PENELITIAN

PERANCANGAN APLIKASI PENGAMANAN CITRA DIGITAL MENGGUNAKAN JAVA DENGAN ALGORITMA RIJNDAEL

Disusun oleh:
Heri Santoso, M.Kom
NIDN. 0119116701

PROGRAM STUDI ILMU KOMPUTER
FAKULTAS SAINS DAN TEKNOLOGI
UIN SUMATERA UTARA MEDAN
2016

PENELITIAN

PERANCANGAN APLIKASI PENGAMANAN CITRA DIGITAL MENGGUNAKAN JAVA DENGAN ALGORITMA RIJNDAEL

Disusun oleh:

Heri Santoso, M.Kom

NIDN. 0119116701

PROGRAM STUDI ILMU KOMPUTER

FAKULTAS SAINS DAN TEKNOLOGI

UIN SUMATERA UTARA MEDAN

2016

REKOMENDASI

Setelah membaca dan menelaah hasil penelitian yang berjudul **"Perancangan Aplikasi Pengamanan Citra Digital Menggunakan Java Dengan Algoritma Rijndael"**. Yang dilakukan oleh Heri Santoso, M.Kom maka saya berkesimpulan bahwa hasil penelitian ini dapat diterima sebagai karya tulis berupa hasil penelitian. Demikianlah rekomendasi diberikan kepada yang bersangkutan untuk dapat dipergunakan sebagaimana mestinya.

Medan, September 2016
Konsultan

Mhd. Furqan, S.Si, MComp. Sc
NIP. 19800806200604 1 003

KATA PENGANTAR

Segala puji bagi Allah SWT yang senantiasa memberikan taburan rahmat dan karunia-NYA sehingga penulis dapat menyelesaikan laporan penelitian yang berjudul : **"PERANCANGAN APLIKASI PENGAMANAN CITRA DIGITAL MENGGUNAKAN JAVA DENGAN ALGORITMA RIJNDAEL".**

Penulisan Laporan Penelitian ini dilakukan dalam rangka melengkapi kewajiban menjadi seorang Dosen dalam melaksanakan Tri Dharma Perguruan Tinggi. Penulis menyadari sepenuhnya bahwa dalam penulisan Penelitian banyak pihak yang membantu dan berpartisipasi. Untuk itu ucapan terima kasih khususnya penulis ucapkan kepada :

1. Bapak Prof. Dr. H. Al Rasyidin, M. Ag selaku Dekan Fakultas Sains dan Teknologi UIN Sumatera Utara Medan.
2. Bapak Mhd Furqan, S.Si., M.Comp.Sc., selaku Ketua Program Studi Ilmu Komputer Fakultas Sains dan Teknologi UIN Sumatera Utara Medan
3. Teman-teman Dosen yang telah membantu pelaksanaan penelitian ini.
4. Teman-teman Staf Laboratorium yang turut membantu atas terselesaikannya penelitian ini.

Atas semua jasa tersebut, penulis serahkan kepada Allah SWT, semoga dibalas dengan Rahmat yang berlipat ganda. Walaupun Penelitian ini telah tersusun dengan sebaik mungkin, penulis tetap mengharapkan kritik dan saran yang membangun untuk penyempurnaan penelitian ini. Semoga penelitian ini dapat berguna bagi kita semua dan bagi penulis sendiri khususnya.

Medan, September 2016
Peneliti,

Heri Santoso, M.Kom

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Citra digital merupakan salah satu data atau informasi yang sering disalahgunakan, oleh karena itu untuk menjaga keamanan dan kerahasiaan suatu data citra digital menjadi hal yang penting. Salah satu pengamanan bisa dilakukan dengan menerapkan algoritma Rijndael. Empat proses utama algoritma ini terdiri dari satu proses permutasi(*ShiftRows*) dan tiga proses subtitusi (*SubBytes, MixColumns*, dan *AddRoundKey*) dan juga proses penjadwalan kunci. Dalam penelitian ini akan dibahas tentang pengamanan citra digital dengan algoritma Rijndael dan juga implementasi algoritma ini dalam mengamankan citra digital. Algoritma Rijndael terdapat dalam proses enkripsi dan dekripsi yang dapat diaplikasikan untuk pengamanan citra digital. Hasil dari aplikasi ini mampu mengenkripsi dan mendekripsi *file* citra tanpa mengubah integritas data dari *file* citra tersebut.

Kata Kunci : Citra Digital, Algoritma Rijndael, Enkripsi, Dekripsi

BAB I

PENDAHULUAN

1.1 Latar Belakang

Teknologi informasi berkembang semakin pesat dan mempengaruhi hampir seluruh aspek kehidupan manusia. Perkembangan tersebut secara langsung maupun tidak langsung mempengaruhi sistem perdagangan, transaksi, bisnis, perbankan, industri dan pemerintahan. Tentunya tingkat keamanan yang tinggi juga semakin diperlukan untuk menghindari penyadapan informasi yang mungkin saja terjadi. Salah satu cara yang bisa digunakan adalah menyandikan (mengenkripsi) informasi atau data rahasia yang akan dikirim, sehingga walaupun pihak yang tidak berkepentingan dapat membaca informasi tersebut tetap sulit bahkan tidak dapat memahami isi informasi tersebut.

Hingga saat ini sistem kriptografi merupakan salah satu solusi untuk menjamin keamanan dari suatu data yaitu dengan menyandikan isi informasi (*plaintext*) tersebut menjadi isi yang sangat sulit bahkan tidak dipahami melalui proses enkripsi dan untuk memperoleh kembali informasi asli dilakukan, proses dekripsi disertai dangan menggunakan kunci yang benar. Melihat penting dan bermanfaatnya teknik enkripsi dan dekripsi, maka akan sangat baik pula jika metode dalam pemrosesannya menggunakan algoritma dengan tingkat keamanan yang sangat tinggi, salah satunya dengan Algoritma Rijndael.

Algoritma Rijndael merupakan salah satu algoritma kriptografi kunci simetris *chiper* blok. Dengan demikian algoritma ini mempergunakan kunci yang

sama saat enkripsi dan dekripsi serta masukan dan keluarannya berupa blok dengan jumlah bit tertentu. Algoritma ini mendukung panjang kunci dan ukuran blok 128-bit sampai 256-bit dengan step 32-bit.

Algoritma Rijndael tidak hanya dapat digunakan untuk mengamankan file dalam bentuk citra digital, Sebelumnya Eko Satria dalam tulisannya yang berjudul Studi Algoritma Rijndael dalam Sistem Keamanan Data telah mengimplementasikan Algoritma Rijndael untuk pengamanan file dalam bentuk teks dengan menggunakan bahasa pemrograman Visual Basic 6.0 .

Aplikasi yang akan dibangun ini nantinya merupakan aplikasi berbasis desktop dengan dukungan bahasa pemrograman Java, dimana Java adalah bahasa pemrograman yang *cross platform*. Berdasarkan uraian di atas, penulis tertarik untuk mengambil judul **" PERANCANGAN APLIKASI PENGAMANAN CITRA DIGITAL MENGGUNAKAN JAVA DENGAN ALGORITMA RIJNDAEL".**

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, terdapat beberapa permasalahan yang dapat dirumuskan sebagai berikut:

1. Bagaimana menerapkan Algoritma Rijndael dalam pengamanan citra digital ?
2. Bagaimana merancang aplikasi kriptografi dengan Algoritma Rijndael untuk mengamankan citra digital ?

1.3 Batasan Masalah

Berdasarkan dengan latar belakang dan perumusan masalah yang telah di uraikan, agar pembahasan dalam penelitian ini tidak meluas, dibatasi hal-hal sebagai berikut:

1. Penggunaan Algoritma Rijndael pada pembangkitan kunci, skema enkripsi dan dekripsi.
2. Jenis file yang digunakan adalah citra digital dengan format file citra JPEG/JPG, PNG dan GIF.
3. Bahasa Pemrograman yang digunakan adalah Java.

1.4 Tujuan Penelitian

Tujuan penelitian ini sebagai berikut:

1. Merancang dan membuat aplikasi pengamanan citra digital dengan menggunakan Algoritma Rijndael.
2. Mengimplementasikan Algoritma Rijndael dengan panjang kunci 128 bit dalam proses enkripsi dan dekripsi pada pengamanan *file* citra digital.

1.5 Manfaat Penelitian

Adapun manfaat dari penelitian ini adalah sebagai berikut:

1. Mampu mengatasi masalah keamanan data, difokuskan pada citra digital.

2. Menambah pengetahuan dan wawasan penulis tentang kriptografi khususnya dalam pengamanan dan kerahasiaan suatu data menggunakan Algoritma Rijndael.

1.6 Metode Penelitian

Adapun metode penelitian yang digunakan dalam penelitian ini adalah sebagai berikut:

1. Studi Kepustakaan

   Studi ini dilakukan dengan cara mencari sekaligus mempelajari beberapa literatur dan artikel mengenai steganografi sebagai acuan dalam perencanaan dan pelaksanaan penelitian ini.

2. Perancangan dan pembuatan Program

   Diperlukan untuk melakukan percobaan dengan program komputer.

3. Pengujian Perangkat Lunak

   Sistem ini akan diuji coba menggunakan metode dan perancangan aplikasi pengamanan citra digital.

4. Pembuatan Laporan

   Membuat laporan tertulis mengenai penelitian ini.

1.7 Sistematika Penulisan

Untuk terarahnya penulisan ini dan untuk menghindari agar tidak terjadinya pembahasan yang berulang serta mempermudah pembaca dalam memahami, maka sistematika penulisannya sebagai berikut :

BAB I : PENDAHULUAN

Pada bab ini dibahas mengenai Latar belakang, perumusan masalah, batasan masalah, tujuan penulisan, manfaat penulisan penelitian dan sistematika penulisan.

BAB II : TINJAUAN PUSTAKA

Membahas tentang tinjauan pustaka dan teori-teori yang menjelaskan beberapa pengertian, konsep dasar serta beberapa hal yang berhubungan dengan judul yang diangkat oleh penulis.

BAB III : ANALISIS DAN PERANCANGAN SISTEM

Bab ini berisi perencanaan kebutuhan, tahap analisis dan tahap desain beserta aksi yang di perlukan dalam setiap tahap.

BAB IV : HASIL DAN PEMBAHASAN

Bab ini berisi implementasi dari hasil pengolahan data yang dilakukan.

BAB V : KESIMPULAN DAN SARAN

Berisikan kesimpulan dan saran dari para penulis.

BAB II

LANDASAN TEORI

2.1 Citra Digital

Citra adalah suatu representasi (gambaran), kemiripan, atau imitasi dari suatu objek. Citra terbagi 2 yaitu ada citra bersifat analog dan ada citra yang bersifat digital. Citra analog adalah citra yang bersifat kontinu seperti gambar pada monitor televisi, foto sinar X, hasil CT Scan dll, sedangkan pada citra digital adalah citra yang dapat diolah oleh komputer (AR Seftiani, 2012)

Sebuah citra digital dapat mewakili oleh sebuah matriks yang terdiri dari M kolom N baris, dimana perpotongan antara kolom dan baris disebut piksel (piksel = *picture element*), yaitu elemen terkecil dari sebuah citra. Piksel mempunyai dua parameter yaitu koordinat dan intensitas atau warna. Nilai yang terdapat pada koordinat (x,y) adalah f(x,y), yaitu besar intensitas atau warna dari piksel di titik itu. Oleh sebab itu, sebuah citra digital dapat di tulis dalam bentuk matriks berikut:

$$f(x,y)= \begin{pmatrix} f(0,0) & f(0,1) & \ldots & f(0,M-1) \\ f(1,0) & \ldots & \ldots & f(1,M-1) \\ \ldots & \ldots & \ldots & \ldots \\ f(N-1,0) & f(N-1,1) & \ldots & f(N-1,M-1) \end{pmatrix}$$

Berdasarkan gambaran tersebut, secara matematis citra digital dapat dituliskan sebagai fungsi intensitas f(x,y), dimana harga x (baris) dan y (kolom)

merupakan koordinat posisi dan f(x,y) adalah nilai fungsi pada setiap titik (x,y) yang menyatakan besar intensitas citra atau tingkat keabuan atau warna dari piksel di titik tersebut. Pada proses digitalisasi (sampling dan kuantitas) diperoleh besar baris M dan kolom N hingga citra membentuk matriks ***M x N*** dan jumlah tingkat keabuan piksel G.

Ada dua jenis format file citra yang sering digunakan dalam pengolahan citra, yaitu citra *bitmap* dan citra *vector*. Citra *bitmap* ini menyimpan data kode citra secara digital dan lengkap (cara penyimpanannya adalah per piksel). Sedangkan pada format file citra vektor merupakan citra vektor yang dihasilkan dari perhitungan matematis dan tidak terdapat piksel, yaitu data yang tersimpan dalam bentuk vektor posisi, dimana yang tersimpan hanya informasi vektor posisi dengan bentuk sebuah fungsi. Contoh format file citra antara lain adalah BMP, GIFF, TIF, JPG/JPEG, IMG, dll.

2.1.1 Format *File* Citra JPEG/JPG

Joint Photographic Experts (JPEG, dibaca *jay peg*) dirancang untuk kompresi citra berwarna (*true color*) atau aras keabuan (*gray scale*) dari suatu citra yang asli, seperti pemandangan asli di dunia ini. JPEG bekerja dengan baik pada *continuous tone images* seperti fotografi atau semua pekerjaan seni yang menginginkan sesuatu terlihat lebih nyata. Tetapi JPEG tidak terlalu bagus pada ketajaman citra dan seni pewarnaan seperti penulisan, kartun yang sederhana atau citra yang menggunakan banyak garis. JPEG mendukung untuk kedalaman warna

24-bit atau sama dengan 16,7 juta warna (2^{24}= 16.777.216 warna). (Edy Sujatmiko, 2007).

2.1.2 Format *File* Citra PNG

Portable Network Graphic (PNG, dibaca *ping*) yang diprakarsai oleh Thomas Boutell dari PNG Development Group pada 1 Oktober 1996 dirancang agar menjadi lebih baikdaripada format yang terdahulu yang sudah dilegalkan yaitu GIF. PNG mempunyai beberapa persamaan fitur dengan GIF antara lain menggunakan kedalaman warna sampai 8-bit piksel (2^8 = 256 warna) / citra berindeks (*indexed images*), mendukung untuk *web browser*, mendukung *transparency*, dan mendukung *interlancing*. Perbedaannya adalah terdapat adanya penambahan fitur-fitur baru antara lain: mendukung kedalaman warna 16-bit (aras keabuan/ *grayscale*) serta citra berwarna 24-bit dan 48-bit (true color images), memiliki *alpha channel* untuk mengontrol *transparency*, memliki *gamma storage* dan *gamma correction* (control untuk ketajaman / *brightness* sebuah citra), memiliki pendeteksi kesalahan (*error detection*), dan memiliki teknik pencocokan warna yang lebih canggih dan akurat. (Edy Sujatmiko, 2007).

2.1.3 Format *File* Citra GIF

Graphics Interchange Format (GIF, dibaca *jiff*, tetapi kebanyakan orang menyebutnya dengan *gif*) dibuat oleh CompuServe Incorporated pada tahun 1987 untuk menyimpan berbagai citra dengan format BMP menjadi citra lain yang mudah untuk diubah pada jaringan komputer. GIF adalah format citra yang paling tua pada *web*, dan begitu dekatnya format citra ini dengan *web* pada saat itu

sehingga para *browser* banyak yang menggunakan format ini. GIF mendukung kedalaman warna hanya sampai 8- bit piksel (2^8 = 256 warna), menggunakan 4 langkah *interlancing* yang memungkinkan sebuah citra dapat diproses secara utuh dengan kualitas yang ditampilkan secar bertahap, mendukung *transparency* (warna latar belakang dapat dibuat transparan), dan mampu menyimpan banyak citra dalam 1 berkas (*multiple images*). GIF termasuk format citra berindeks (*indexed images*) karena hanya bisa mendukung sampai 256 warna. (Edy Sujatmiko, 2007).

2.2 Kriptografi

Kriptografi adalah ilmu dan seni untuk menjaga kemananan pesan (Dony Ariyus. 2008). Dalam dunia kriptografi, pesan disebut *plaintext* atau *cleartext*. Proses untuk menyamarkan pesan dengan cara sedemikian rupa untuk menyembunyikan isi aslinya disebut enkripsi. Pesan yang telah dienkripsi disebut *ciphertext*. Proses pengembalian sebuah *ciphertext* ke *plaintext* disebut dekripsi.

Cryptographer adalah orang yang mempraktekkan ilmu kriptografi, sedangkan *Cryptanalyst* adalah orang yang mempraktekkan, *Cryptanalysis* adalah seni dan ilmu dalam memecahkan *ciphertext*.

Ada empat tujuan mendasar dari ilmu kriptografi yang juga merupakan aspek keamanan informasi yaitu (Dony Ariyus, 2008) :

1. Kerahasiaan (*confidentiality*), adalah layanan yang digunakan untuk menjaga isi dari informasi dari siapapun kecuali yang memiliki otoritas atau kunci rahasia untuk membuka informasi yang telah disandi.
2. Integritas data (*data integrity*), adalah berhubungan dengan penjagaan dari perubahan data secara tidak sah.
3. Otentikasi (authentication), adalah berhubungan dengan identifikasi atau pengenalan, baik secara kesatuan sistem maupun informasi itu sendiri.
4. Nirpenyangkal (*non-repudiation*), adalah usaha untuk mencegah terjadinya penyangkalan terhadap pengiriman atau terciptanya suatu informasi oleh yang mengirimkan atau yang membuat.

2.2.1 Algoritma Kriptografi

Algoritma kriptografi disebut juga *chiper* yaitu aturan untuk *enchipering* dan *dechipering*, atau fungsi matematika yang digunakan untuk enkripsi dan dekripsi. Beberapa *chiper* memerlukan algoritma yang berbeda untuk *enciphering* dan *dechipering*.

Algoritma kriptografi terdiri dari fungsi dasar yaitu (Dony Ariyus, 2008):

1. Enkripsi

   Enkripsi merupakan pengamanan data dengan cara mengubah pesan asli (*Plaintext*) menjadi kode-kode yang tidak dapat dimengerti (*Chipertext*).

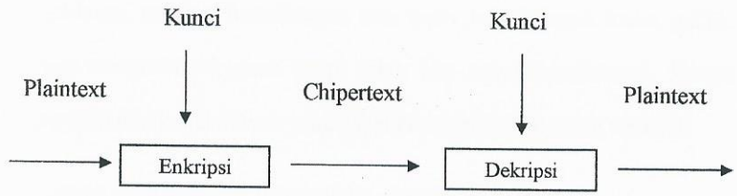
2. Dekripsi

   Dekripsi merupakan kebalikan dari enkripsi, yaitu pesan yang telah dienkripsi dikembalikan ke bentuk asalnya.

3. Kunci

   Kunci yang dipakai untuk enkripsi dan dekripsi. Kunci terbagi 2 yaitu kunci pribadi (*private key*) dan kunci umum (*public key*).

Gambar 2.1 memperlihatkan skema enkripsi dan dekripsi dengan menggunakan kunci.



Gambar 2.1 Proses Enkripsi dan Dekripsi

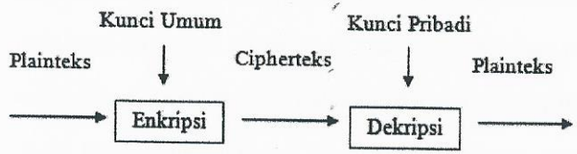
(**Sumber:** Muhamad Farid Fachrurozi, 2010)

Algoritma kriptografi dapat diklasifikasikan menjadi dua jenis berdasarkan kuncinya yaitu algoritma Asimetri dan algoritma Simetri.

2.2.2 Algoritma Kunci Asimetri

Algoritma kunci asimetri sering juga disebut dengan algoritma kunci publik, dengan arti kata kunci yang digunakan melakukan enkripsi dan dekripsi berbeda (Dony Ariyus, 2008). Pada algoritma asimetri kunci terbagi menjadi dua bagian, yaitu:

1. Kunci umum (*public key*), kunci yang boleh semua orang tahu (dipublikasikan).
2. Kunci rahasia (*private key*), kunci yang dirahasiakan (hanya boleh diketahui oleh satu orang.



Gambar 2.2 Algoritma Kunci Asimetri
(**Sumber:** Dony Arius, 2008)

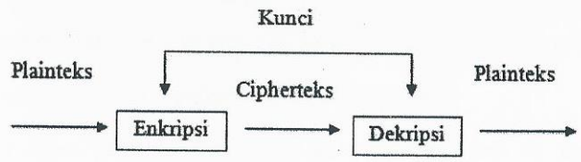
Kunci-kunci tersebut berhubungan satu sama lain. Dengan kunci publik orang dapat mengenkripsi pesan tetapi tidak bisa mendekripsikannya. Hanya orang yang memiliki kunci rahasia yang dapat mendekripsikan pesan tersebut.

Algoritma yang memakai kunci publik diantaranya adalah :

1. *Digital Signature Algorithm* (DSA),
2. *RSA*,
3. *Diffle-Hellman* (DH),
4. *Elliptic Curve Cryptography* (ECC),
5. Kriptografi Quantum dan lain sebagainya.

2.2.3 Algoritma Kunci Simetri

Algoritma ini sering disebut dengan algoritma klasik karena memakai kunci yang sama untuk kegiatan enkripsi maupun dekripsi (Dony Ariyus, 2008).



Gambar 2.3 Algoritma Kunci Simetri
(**Sumber:** Dony Arius, 2008)

Bila mengirim pesan dengan menggunakan algoritma ini, si penerima pesan harus diberitahu kunci dari pesan tersebut agar bisa mendekripsikan pesan yang terkirim. Keamanan dari pesan yang menggunakan algoritma ini tergantung pada kunci. Jika kunci tersebut diketahui oleh orang lain maka orang tersebut akan dapat melakukan enkripsi dan dekripsi terhadap pesan. Algoritma yang memakai kunci simetri diantaranya adalah:

1. *Data Encryption Standard* (DES),
2. RC2, RC4, RC5, RC6,
3. *International Data Encryption Algorithm* (IDEA),
4. *Advanced Encryption Standard* (AES),
5. *On Time Pad* (OTP),
6. A5, dan lain sebagainya.

2.2.4 Tipe dan Mode Algoritma Kunci Simetri

Algoritma kunci simetri memiliki dua tipe yang umum digunakan, adapun tipe dan mode algoritma kunci simetri adalah (Dony Ariyus, 2008):

1. *Chiper* blok, yang beroperasi pada blok-blok plainteks. Ukuran blok dapat bermacam-macam tergantung pada algoritma enkripsi yang digunakan, biasanya berkisar anatar 64 atau 128 bit. Hasil enkripsi dari suatu *plaintext* yang sama, akan menghasilkan *chipertext* yang sama.
2. *Chiper* aliran, yang beroperasi pada aliran *plaintext* satu bit atau byte setiap waktu. Hasil enkripsi dari suatu *plaintext* yang sama belum tentu menghasilkan *chipertext* yang sama.

2.3 Algoritma Rijndael

Rijndael adalah Algoritma kriptografi yang didesain oleh oleh Vincent Rijmen dan John Daemen asal Belgia keluar sebagai pemenang kontes algoritma kriptografi pengganti DES yang diadakan oleh NIST (National Institutes of Standards and Technology) milik pemerintah Amerika Serikat pada 26 November 2001 (Dony Ariyus, 2008). Algoritma Rijndael inilah yang kemudian dikenal dengan Advanced Encryption Standard (AES). Setelah mengalami beberapa proses standardisasi oleh NIST, Rijndael kemudian diadopsi menjadi standard algoritma kriptografi secara resmi pada 22 Mei 2002. Pada 2006, AES merupakan salah satu algoritma terpopuler yang digunakan dalam kriptografi kunci simetrik.

Algoritma Rijndael menggunakan subtitusi, permutasi dan sejumlah putaran yang dikenakan pada tiap blok yang akan dienkripsi dan didekripsi. Untuk setiap putarannya, Rijndael menggunakan kunci yang berbeda. Kunci setiap putaran disebut *round key*. Rijndael beroperasi dalam orientasi *byte* sehingga memungkinkan untuk implementasi algoritma yang efisien ke dalam *software* dan *hardware*. Ukuran blok untuk algoritma Rijndael adalah 128 bit (16 *byte*).

Rijndael mendukung panjang kunci 128 bit sampai 256 bit dengan step 32 bit. Panjang kunci dan ukuran blok dapat dipilih secara independen. Setiap blok dienkripsi dalam sejumlah putaran tertentu, sebgaimana halnya pada DES. Karena AES menetapkan panjang kunci adalah 128, 192 dan 256 bit, maka dikenal AES-128, AES-192, dan AES-256.

Tabel 2.1 Perbedaan Kunci Rijndael

	Panjang Kunci (Nk)	Ukuran Blok (Nb)	Jumlah *Round* (Nr)
AES 128	4	4	10
AES 192	6	4	12
AES 256	8	4	14

(**Sumber:** Dony Arius, 2008)

Algoritma Rijndael mempunyai 3 (tiga) parameter (Dony Ariyus,2008):

1. *Plaintext* adalah *array* yang berukuran 16-*byte*, yang berisi data masukkan.
2. *Chipertext* adalah *array* yang berukuran 16-*byte*, yang berisi hasil enkripsi.
3. Kunci adalah *array* yang berukuran 16-*byte*, yang berisi kunci *chiper* (disebut juga *chiper key*).

Dengan 16 *byte*, maka baik blok data maupun kunci yang berukuran 128 bit dapat disimpan didalam ketiga *array* tersebut (128= 16 x 8).

Selama kalkulasi *plaintext* menjadi *chipertext*, status sekarang dari data disimpan didalam *array of byte* dua dimensi, *state* yang berukuran NROWS x NCOLS. Elemen *array state* diacu sebagai S[r,c], dengan $0 \leq r < 4$ dan $0 \leq c <$ Nc (Nc adalah panjang blok dibagi 32). Pada AES, Nc = 128/32 = 4.

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

Gambar 2.4 Ilustrasi *Array State*

(**Sumber:** Muhamad Farid Fachrurozi, 2006)

Tiap elemen *array state* diisi dengan 8 bit teks (1 *byte*) dalam notasi HEX. Urutan pengisian dimulai dari kolom awal (c=0) sampai kolom terakhir (pada contoh diatas, c=3) dan dari baris awal (r=0) sampai baris akhir (s=3). Pada Gambar 5 , 1 *byte* teks pertama disimpan dalam notasi HEX pada $S_{0,0}$ 1 byte teks kedua pada $S_{1,0}$ 1 *byte* teks kelima pada $S_{0,1}$ dan seterusnya.

19	A0	9A	E9
3D	F4	06	F8
E3	E2	8D	48
BE	2B	2A	08

Gambar 2.5 Ilustrasi Pengisian *Array State*

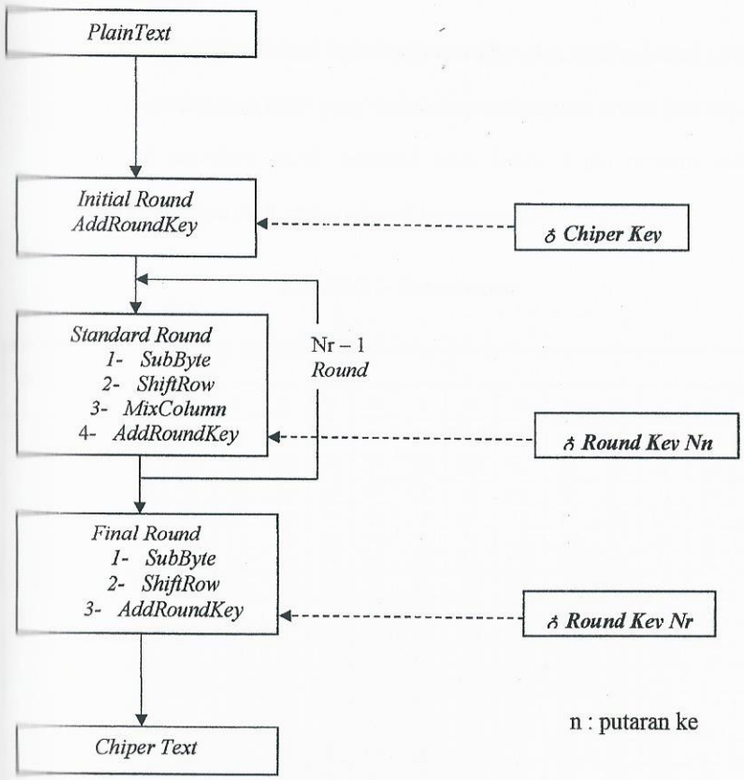
(**Sumber:** Elfira Yolanda S , 2008)

19 = 00011001 (1 *byte* pertama)

3D = 00111101 (1 *byte* kedua),dst

Bagian selanjutnya akan menjelaskan lebih lanjut perlakuan perlakuan yang dikenakan pada *array state* tersebut dari awal yang berisi salinan *plaintext* sampai menghasilkan *chipertext.*

2.3.1 Proses Enkripsi Algoritma Rijndael (AES)



Gambar 2.6 Diagram Proses Enkripsi Algoritma Rijndael
(**Sumber:** Elfira Yolanda S , 2008)

Secara garis besar proses enkripsi algoritma Rijndael yang beroperasi pada blok 128 bit dengan kunci 128 bit adalah sebagai berikut (Dony Ariyus, 2008):

1. *AddRoundKey,* melakukan *XOR* antara *state* awal (*plaintext*) dengan *chiper key*. Tahap ini disebut juga *initial round.*

2. Putaran sebanyak Nr-1 kali. Proses yang dilakukan pada setiap putaran adalah :

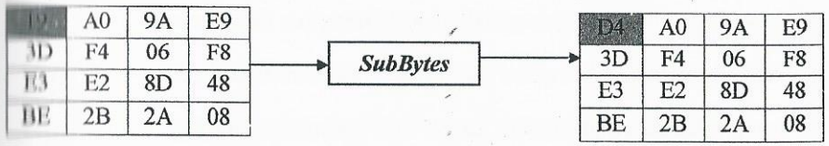
- *SubBytes* adalah subtitusi *byte* dengan mengunakan tabel subtitusi (*S-Box*). Dua digit bilangan HEX yang merupakan representasi 1 *byte* dari tiap teks menjadi koordinat untuk subsitusi pada S-*box*. Digit pertama sebagai koordinat x dan digit kedua sebagai koordinat y.

Tabel 2.2 *S-Box* Rijndael

HEX		Y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
X	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	3	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

(**Sumber:** Dony Arius, 2008)

Ilustrasi proses *SubBytes* pada elemen pertama dari *array state* yaitu 19 disubtitusikan menggunakan Tabel 2.2 *S-Box*, dengan cara memasukkan nilai x – 1 dan nilai y = 9 hingga mendapatkan hasil D4. Proses ini dilakukan untuk tiap elemen pada *array state* hingga mendapatkan hasil.



Gambar 2.7 Ilustrasi Transformasi *SubBytes*

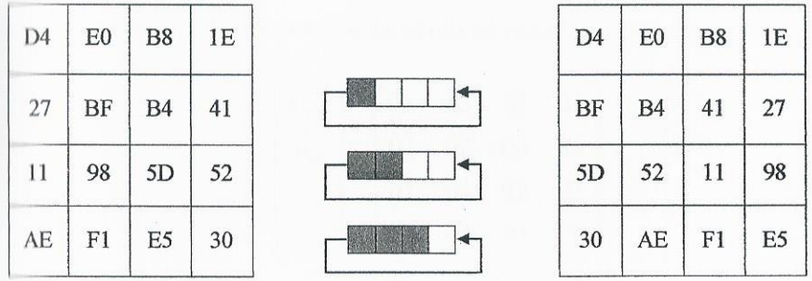
(**Sumber:** Elfira Yolanda S , 2008)

D4	E0	B8	1E
27	BF	B4	41
11	98	5D	52
AE	F1	E5	30

Gambar 2.8 Hasil Transformasi *SubBytes*

(**Sumber:** Elfira Yolanda S , 2008)

- *ShiftRows* adalah pergeseran baris baris *array state* secara *wrapping*, pada 3 (tiga) baris terakhir dari array state. Jumlah pergeseran bergantung pada nilai baris r. Baris r=1 digeser sejauh 1 byte, baris r=2 digeser sejauh 2 byte dan baris r=3 digeser sejauh 3 byte. Baris r=0 tidak digeser.



Gambar 2.9 Ilustrasi Transformasi *ShiftRow*

(**Sumber:** Elfira Yolanda S , 2008)

- *MixColumns* adalah mengacak data di masing-masing kolom *array state*. Pengacakan dilakukan dengan mengalikan setiap kolom dari *array state* dengan polinom a(x)mod(x^4+1). Setiap kolom diperlakukan sebagai polinom 4-suku pada GF(2^8). Polinom a(x)y\ yang ditetapkan adalah :

$$a(x)=\{03\}x^3+\{01\}x^2+\{01\}x+\{02\}$$

Tranformasi ini dinyatakan sebagai perkalian matriks:

$$s'(x)=a(x) \otimes s(x)$$

$$s'_{0,c}=(\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$

$$s'_{1,c}=s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$$

$$s'_{2,c}=s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{3,c})$$

$$s'_{3,c}=(\{03\} \bullet s_{0,c}) \oplus s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{3,c})$$

Ilustrasi perkalian matriks dapat dilihat pada Gambar 2.10

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

Gambar 2.10 Ilustrasi Perkalian Matriks *MixColumn*
(**Sumber:** Elfira Yolanda S , 2008)

Ilustrasi *MixColumn* dapat dilihat pada Gambar 2.10 . Hasil proses sebelumnya ditunjukkan pada Gambar 2.11 a . Untuk tiap kolom dari hasil tersebut dilakukan perkalian matriks seperti pada Gambar 2.11 b. Setelah *MixColumn* dilakukan pada keempat kolom *array state*, didapatkan hasil pada Gambar 2.12 .

D4	E0	B8	1E
BF	B4	41	27
5D	52	11	98
30	AE	F1	E5

a

D4		02	01	01	03		04
BF		03	02	01	01		66
5D	●	01	03	02	01	=	81
30		01	01	02	03		E5

b

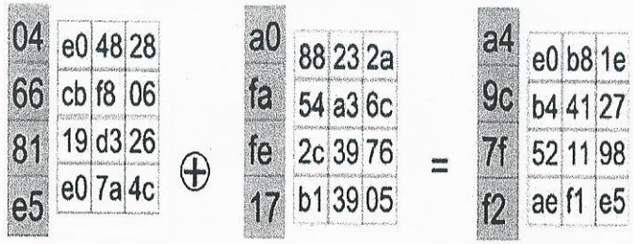
Gambar 2.11 Ilustrasi Transformasi *MixColumn*

(**Sumber:** Elfira Yolanda S , 2008)

04	E0	48	28
66	CB	F8	06
81	19	D3	26
E5	9A	7A	4C

Gambar 2.12 Hasil Transformasi *MixColumn*

- *AddRoundKey* adalah melakukan *XOR* antara *state* sekarang *round key*. Ilustrasi *AddRoundkey* dapat dilihat pada Gambar 2.13. Tiap kolom dari *array state* dikenakan *XOR* dengan kolom *round* yang sepadan. Proses ini dilakukan terhadap seluruh kolom *array state*. Contoh hasil proses *AddRoundkey* yang lengkap dapat dilihat pada Gambar 2.14 .



Keterangan :
ô Operasi *XOR*

Gambar 2.13 Ilustrasi Transformasi *AddRoundKey*
(**Sumber:** Elfira Yolanda S , 2008)

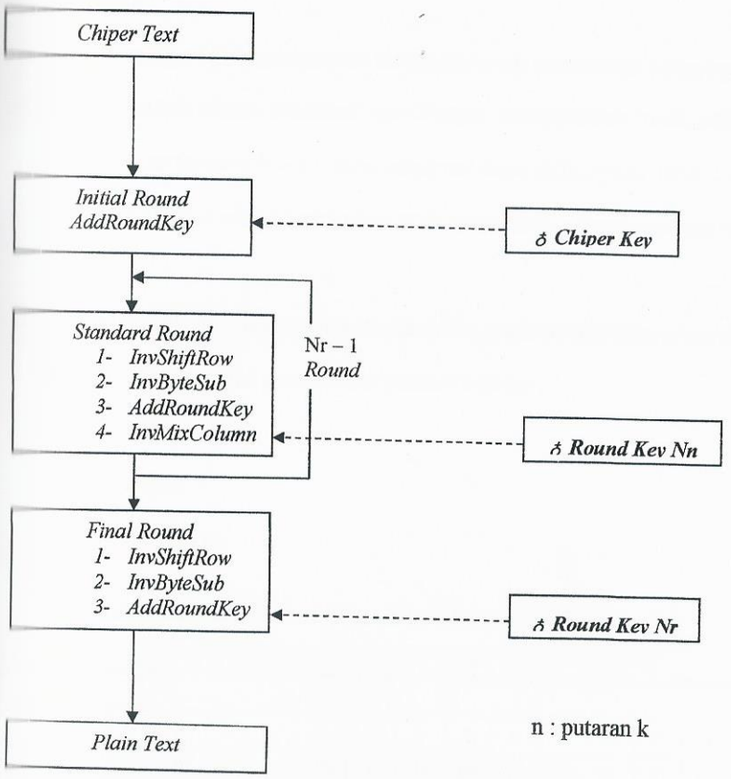
A4	68	6B	02
9C	9F	5B	6A
7F	35	EA	50
F2	2B	43	49

Gambar 2.14 Hasil Transformasi *AddRoundKey*

3. *Final Round*, proses untuk putaran terakhir :

- *SubBytes*
- *ShiftRows*
- *AddRoundKey*

2.3.2 Proses Dekripsi Algoritma Rijndael (AES)



Gambar 2.15 Diagram Proses Dekripsi Algoritma Rijndael
(**Sumber:** Elfira Yolanda S , 2008)

Secara garis besar proses dekripsi algoritma Rijndael yang beroperasi pada blok 128 bit dengan kunci 128 bit adalah berikut:

1. *AddRoundKey*, melakukan *XOR* antara *state* awal (*plaintext*) dengan *chiper key*. Tahap ini disebut juga *initial round*.

2. Putaran sebanyak Nr-1 kali. Proses yang dilakukan pada setiap putaran adalah :

- *InvShiftRow* adalah pergeseran baris-baris *array state* secara *wrapping*.
- *InvByteSub* adalah substitusi *byte* dengan menggunakan tabel subtitusi kebalikan (*inverse S-box*). Tabel substitusi dapat dilihat pada Tabel 2.3.
- *AddRoundKey* adalah melakukan *XOR* antara *state* sekarang dengan *round key*.
- *InvMixColumn* adalah mengacak data dimasing-masing kolom *array state*.

3. *Final round* adalah proses untuk putaran terakhir :

- *InvShiftRow*
- *InvByteSub*
- *AddRoundKey*

Tabel 2.3 Tabel *inverse S-Box* dalam Transformasi *InvByteSub*

HEX		Y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
X	0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
	1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
	2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
	3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
	4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
	5	6C	70	48	50	FD	ED	B9	DA	5E	A5	46	57	A7	8D	9D	84
	6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
	7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	A3	8A	6B
	8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
	9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
	A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
	B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
	C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
	D	60	51	71	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
	E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
	F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

(**Sumber:** Dony Arius, 2008)

2.3.3 Ekspansi Kunci

Algoritma Rijndael melaksanakan *chiper key* dan membuat suatu ekspansi kunci untuk menghasilkan suatu *key schedule*. Jika ekspansi kunci yang diperlukan Rijndael Nb(Nr+1) word, sehingga bisa digunakan AES 128 bit, maka 4(10+1)=40 word=44x32 bit=1408 bit *subkey*. Ekspansi dari 128 menjadi 1408 bit *subkey*, proses ini disebut dengan *key schedule*. *Subkey* ini diperlukan karena setiap *round* merupakan suatu inisial dari Nb *word* untuk Nr=10, dari operasi ini akan didapatkan *schedule* kunci yang berisi *array* linier 4 *byte word*(w_i), 0=i(Nr+1).

2.4 Rekayasa Perangkat Lunak (*Software Engineering*)

Rekayasa perangkat Lunak merupakan pembangunan dengan menggunakan prinsip atau konsep rekayasa dengan tujuan menghasilkan perangkat lunak yang bernilai ekonomi yang dipercaya dan bekerja secara efisien menggunakan mesin. Perangkat lunak banyak dibuat dan pada akhirnya sering tidak digunakan karena tidak memenuhi kebutuhan pelanggan atau bahkan karena masalah non-teknis seperti keenggan pemakai perangkat lunak (*user*) untuk mengubah cara kerja dari manual ke otomatis atau ketidakmampuan *user* menggunakan komputer. Oleh karena itu rekayasa perangkat lunak dibutuhkan agar perangkat lunak yang dibuat tidak hanya menjadi perangkat lunak yang tidak terpakai. (Rosa A.S dan M. Shalahuddin, 2014)

2.5 Analisis dan Perancangan Sistem

Analisis sistem didefenisikan sebagai bagaimana memahami dan menspesifikasi dengan detail apa yang harus dilakukan oleh sistem, sedangkan sistem desain diartikan sebagai menjelaskan detail bagaimana bagian-bagian dari sistem informasi diimplementasikan, sehingga analisis dan desain sistem informasi (ANSI) bisa didefenisikan sebagai proses organisasional kompleks dimana sistem informasi berbasis komputer diimplementasikan, sehingga bisa diringkas sebagai berikut (Hanif Al Fatta, 2007) :

Analysis : mendefenisikan masalah

- *From requitments to specification*

Design : memecahkan masalah

- *From specification to implementation*

2.6 UML (*Unified Modeling Language)*

UML *(Unified Modeling Language)* adalah bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek. UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun dan dokumentasi dari sistem perangkat lunak. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung.UML hanya berfungsi untuk melakukan pemodelan. Jadi penggunaan UML tidak terbatas pada metodologi tertentu,

meskipun pada kenyataannya UML paling banyak digunakan pada metodo;ogi berorientasi objek. (A.S, Rosa dan M. Shalahuddin, 2014)

Bagian-bagian utama dari UML adalah *view*, diagram, model *element* dan *general mechanism*.

1. *View*

   View digunakan untuk melihat sistem yang dimodelkan dari beberapa aspek yang berbeda. *View* bukan melihat grafik, tapi merupakan suatu abstraksi yang berisi sejumlah diagram. Beberapa jenis *view* dalam UML antara lain : *use case view*, *logical view*, component *view*, *concurrency view*, dan *deployment view*.

2. Diagram

   Diagram berbentuk grafik yang menunjukkan simbol elemen model yang disusun untuk mengilustrasikan bagian atau aspek tertentu dari sistem. Sebuah diagram merupakan bagian dari suatu *view* tertentu dan ketika digambarkan biasanya dialokasikan untuk *view* tertentu. Adapun jenis diagram antara lain :

   - *Use Case* Diagram

     Usecase atau diagram *usecase* merupakan pemodelan untuk melakukan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan

untuk mengetahui fungsi apa saja yang ada di dalam sebuah system informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu. syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefenisian apa yang disebut aktor dan *use case.*

- Aktor merupakan orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
- *Use Case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

➢ *Activity* Diagram

Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem. Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut:

- Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan
- merupakan proses bisnis sistem yang didefinisikan.

- Urutan atau pengelompokan tampilan dari sistem / *Pengguna interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
- Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah
- pengujian yang perlu didefinisikan kasus ujinya.

Pada dasarnya diagram *activity* sering digunakan oleh *flowchart* Diagram ini berhubungan dengan diagram *statechart*. Diagram *statechart* berfokus pada objek yang dalam suatu proses, d*iagram activity* berfokus pada aktifitas-aktifitas yang terjadi terkait dalam suatu proses tunggal. Jadi dengan kata lain, diagram ini menunjukkan bagaimana aktifitas-aktifitas tersebut tergantung satu sama lain.

2.7 Bahasa Pemrograman Java

Java adalah sebuah bahasa pemrograman beroientasi objek yang dikembangkan oleh *Sun Microsystems*. Menurut definisi dari Sun, Java adalah nama untuk sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer *standalone* ataupun pada lingkungan jaringan.

Untuk beragam aplikasi yang dibuat dengan bahasa Java, java dipaketkan dalam edisi–edisi berikut (Didik Dwi Prasetyo, 2007) :

1. Java 2 *Standard Edition* (J2SE).

Edisi ini ditujukan bagi lingkungan *workstation*, seperti buatan aplikasi-aplikasi dekstop.

2. Java 2 Enterprise Edition (J2EE).

   Edisi ini ditujukan bagi lingkungan internet atau aplikasi distribusi dalam skala besar.

3. Java 2 Micro Edition (J2ME).

   Edisi ini ditujukan bagi lingkungan dengan sumber daya batas, seperti *smartcard*, ponsel dan PDA.

Masing – masing edisi berisi Java 2 Software Development Kit (J2SDK) untuk mengembangkan aplikasi dan Java 2 Runtime Environent (J2RE) untuk menjalankan aplikasi.

Java adalah salah satu bahasa pemrograman berorientasi objek (OOP-*Object Oriented Programming*). Paradigma OOP menyelesaikan masalah dengan merepresentasikan masalah ke model objek.

Keutamaan Java dibanding bahasa pemrograman lain :

1. *Cross platform*, dengan adanya *Java Virtual Machine* (JVM).
2. Pengembangan didukung oleh *programmer* secara luas.
3. *Automatic Garbage Collection*, membebaskan *programmer* dari tugas manajemen memori.

BAB III

ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Masalah

Berdasarkan uraian pada bab sebelumnya, rumusan masalah yang akan dibahas dalam penelitian ini adalah penerapan algoritma Rijndael (AES) dalam proses pengamanan *file* citra. Pada subbab berikut akan dibahas mengenai hal-hal yang diperlukan menyelesaikan masalah tersebut.

3.1.1 Analisis Aplikasi Perangkat Lunak

Perangkat lunak AES Kriptograi pengamanan *file* citra adalah sebuah sistem yang digunakan oleh pribadi atau suatu organisasi untuk mengamankan informasi rahasia. Keamanan sistem ini terletak pada membuka dan menyimpan isi *file* tersebut. proses enkripsi dengan algoritma Rijndael (AES) dilakukan pada saat informasi berupa *file* citra disimpan, sedangkan proses dekripsi dengan algoritma Rijndael (AES) dilakukan pada saat membuka informasi berupa *file* citra.

3.1.2 Analisis Layanan Kriptografi

Pada bab sebelumnya diuraikan mengenai layanan yang diberikan dalam sebuah sistem kriptografi meliputi kerahasiaan, integritas, autentikasi dan nirpenolakan. Dalam penelitian ini, layanan kriptografi yang diberikan yaitu kerahasiaan. Penyandian pada *file* citra bertujuan agar hanya pemilik *file* tersebut

yang berhak yang dapat membuka informasi dari *file* yang sebenarnya, dengan kata lain hanya orang yang memiliki kunci untuk dekripsi yang dapat membuka *file* dengan benar.

3.1.3 Analisis Algoritma Kriptografi

Algoritma kriptografi yang digunakan dalam penelitian ini adalah Rijndael (AES). Alasan pemilihan Rijndael (AES) berdasarkan pertimbangan berikut:

1. Tergolong dalam algoritma kunci simetris, sebuah kunci tunggal dapat digunakan untuk mengenkrip dan mendekrip pesan, hal ini mempermudah kecepatan dalam distribusi kunci. Pengirim dan penerima menyetujui kunci rahasia tanpa ada orang lain yang mengetahuinya dimana dua pihak tersebut dapat berkomunikasi tanpa takut akan disadap. Kunci simetris tidak hanya berkaitan dengan enkripsi tetapi juga berkaitan dengan otentikasi.
2. Rijndael merupakan algoritma pemenang dari kontes AES sebagai pengganti algoritma DES. Algoritma Rijndael memiliki keseimbangan antara keamanan dan fleksibilitas dalam berbagai *platform software* dan *hardware*. Algoritma Rijndael dirancang untuk ketahanan terhadap semua jenis serangan yang diketahui, kesederhanaan rancangan, kekompakan kode dan kecepatan pada berbagai *platform*.

3.1.3.1 Pembangkitan Kunci

Untuk melakukan proses enkripsi dan dekripsi, Rijndael (AES) memerlukan kunci ekspansi *chipper key*. Ekspansi *chipper Key* digunakan untuk membentuk *Round Key* yang akan digunakan pada langkah-langkah enkripsi *plaintext* (*state*). Ekspansi kunci ini memiliki tahapan khusus yang dikenal sebagai Rijndael's *Key schedule*.

Secara umum, Rijndael Key schedule memanfaatkan beberapa operasi :

1. *Rot Word*

   Operasi ini merupakan operasi pemindahan *byte*. *Byte* terujung dipindahkan ke ujung lainnya tanpa mengubah keterurutan komponen lain.

2. *Rcon*

   Operasi ini merupakan eksponensiasi dari bilangan bulat 2 dalam semesta Rijndael *finite field*. *Finite field* tidak akan dibahas secara mendalam pada uraian makalah ini karena kajian mengenai hal tersebut tercakup dalam lingkup aljabar abstrak.

3. *S-box*

   tabel ini bukan merupakan operasi dalam arti *literal*. *Substitution box* merupakan tabel yang menyulih nilai *byte* dengan *round*, sembilan lainnya pada tahapan *round*, serta satu *Round Key* yang digunakan pada *final*

round. Operasi-operasi di atas kemudian diimplementasikan pada *Key schedule* dan diulang sebanyak kebutuhan.

Pada proses pembentukan *Round Key* tersebut nantinya akan digunakan pada proses *AddRoundKey*. Langkah yang dilakukan:

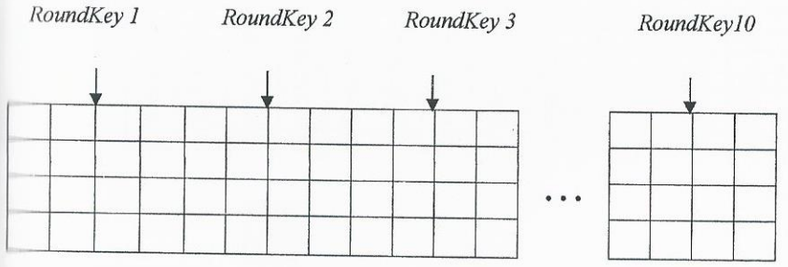
1. Menyediakan *array* kunci berukuran 4x4 yang terdiri dari 128 bit dimana pada tiap selnya terdiri atas 1 *byte*.

   Misal, *array* kuncinya adalah seperti ini:

2B	28	AB	09
7E	AE	F7	CF
15	D2	15	4F
16	A6	88	3C

Gambar 3.1 *Array* Kunci

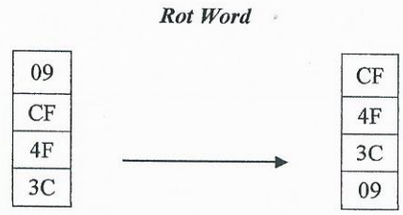
Pada *round key* yang akan terjadi hingga akhir proses terlihat pada Gambar 3.1.



Gambar 3.2 Proses *Array*

2. Melakukan *Rot Word* terhadap kolom terakhir dari *array* kunci diatas. *Rot word* berarti menukarkan posisi baris ke-1 dengan baris ke-4, baris ke-2

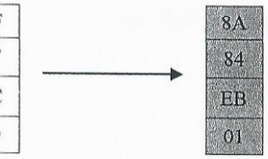
dengan ke-1, baris ke-3 dengan ke-2 dan baris ke-4 dengan ke-3. Lihat Gambar 3.3.



Gambar 3.3 *Rot Word*

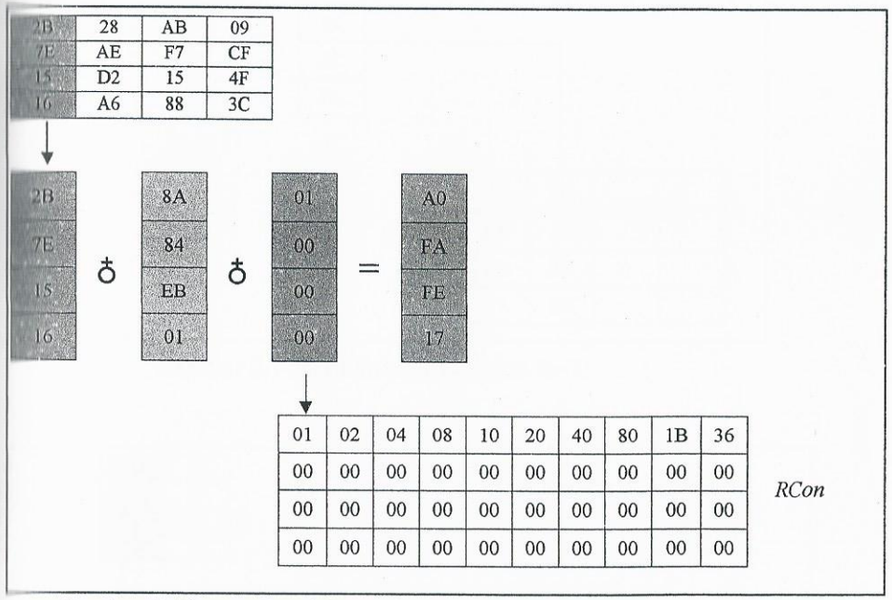
3. Hasil dari *Rot word* akan di *SubBytes* kan dengan sebuah *S-Box* Rijndael (AES). Lihat gambar 3.4.

HEX		Y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
X	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	3	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16



Gambar 3.4 Hasil *SubBytes*

4. Langkah selanjutnya menentukan kolom ke-1 pada *round key* pertama. Lakukan *XOR* antara *chipper key*, hasil *SubBytes* dan *Rcon*. Lihat Gambar 3.5.

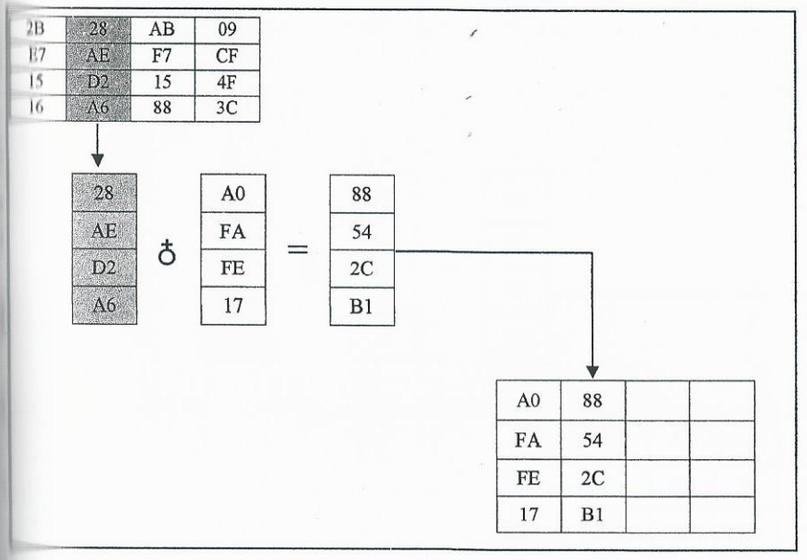


Gambar 3.5 Proses Pengisian Kolom Ke-1 pada *Round Key* Pertama.

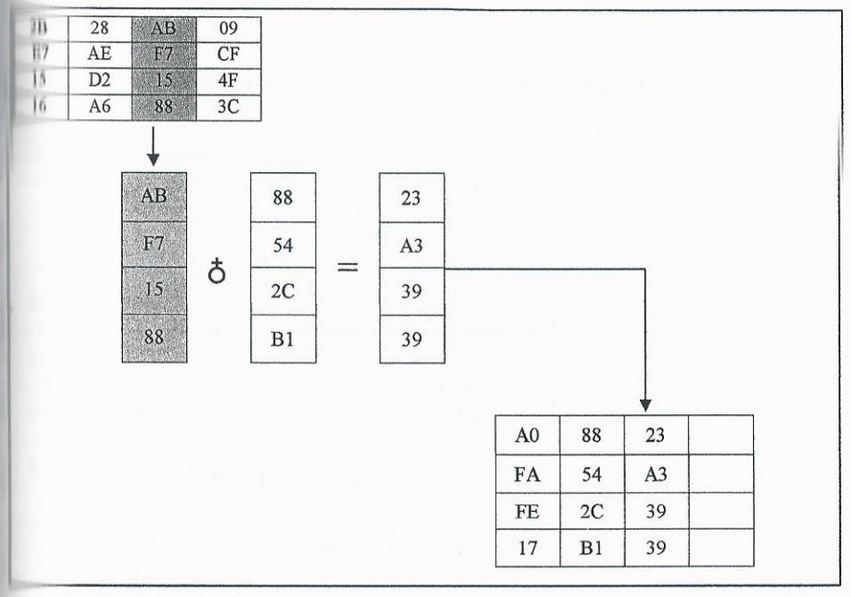
5. Setelah mendapatkan kolom ke-1 pada *round key* pertama seperti gambar 3.6, selanjutnya lakukan pengisian terhadap kolom ke-2, 3 dan 4. Lihat gambar 3.7, gambar 3.8, dan gambar 3.9.

A0			
FA			
FE			
17			

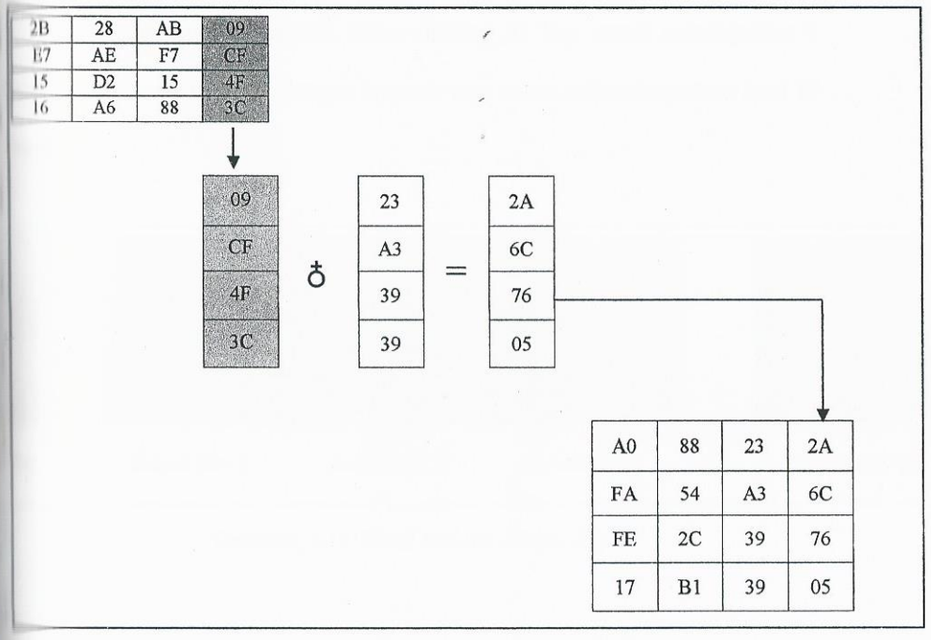
Gambar 3.6 Hasil *Round Key* Kolom ke-1



Gambar 3.7 Hasil *Round Key* kolom ke-2

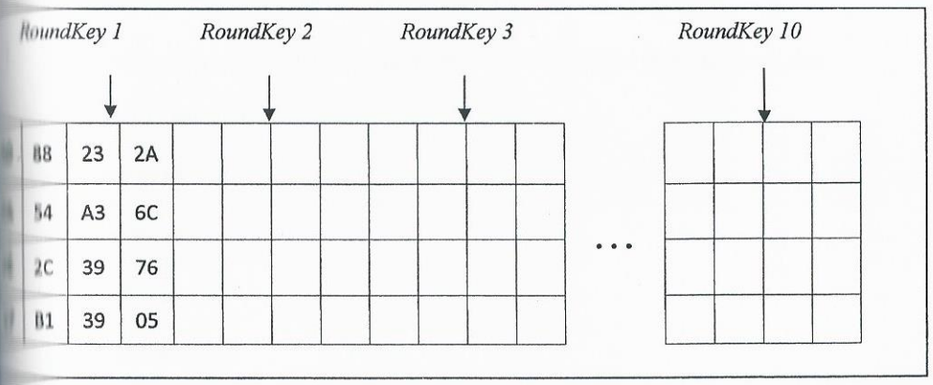


Gambar 3.8 Hasil *Round Key* Kolom Ke-3



Gambar 3.9 Hasil *Round Key* Kolom Ke-4

Dari hasil tersebut, maka telah didapatkan 4 kolom pada *round key* pertama.



Gambar 3.10 Hasil *Round Key* 1

Kemudian langkah-langkah diatas diulang 9x lagi untuk mendapatkan 9 buah *round key* berikutnya. Dengan langkah yang sama, maka didapatkan hasil 10 *round key*.

AB	09	A0	88	23	2A	F2	7A	23	73	3D	47	1E	6D	...	D0	C9	E1	B6
F7	CF	FA	54	A3	6C	C2	96	A3	59	80	16	23	7A		14	EE	3F	63
15	4F	FE	2C	39	76	95	B9	39	F6	47	FE	7E	88		F9	25	0C	0C
88	3C	17	B1	39	05	F2	43	39	7F	7D	3E	44	3B		A8	89	C8	A6
Key		*Round Key* 1				*Round Key* 2				*Round Key* 3					*Round Key* 10			

Gambar 3.11 Hasil seluruh *Round Key*

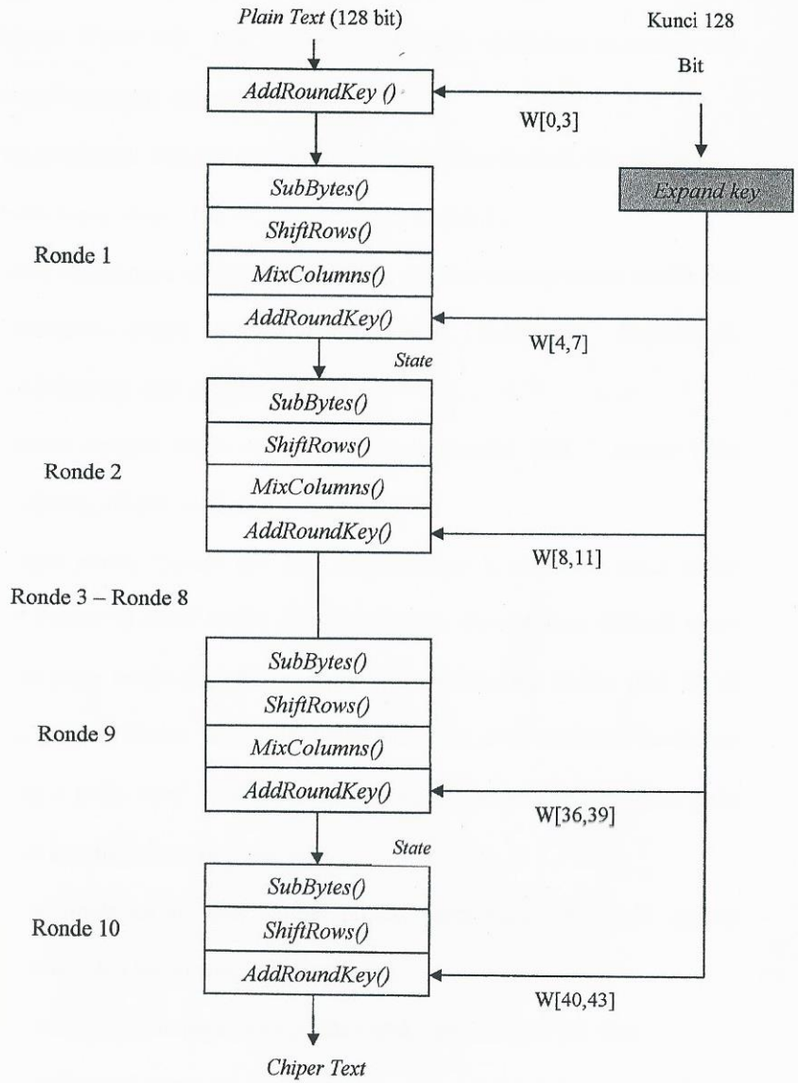
3.1.3.2 Proses Enkripsi

Berdasarkan pembahasan pada BAB sebelumnya tahapan enkripsi pada algoritma Rijndael (AES) sudah dijelaskan sebelumnya.

Adapun urutan proses enkripsi algoritma Rijndael dengan panjang kunci dan ukuran blok 128 bit adalah:

- *SubBytes()*
- *Shift Rows()*
- *Mix Columns()*
- *AddRoundKey()*

Berikut penjelasan proses enkripsi melalui diagram :



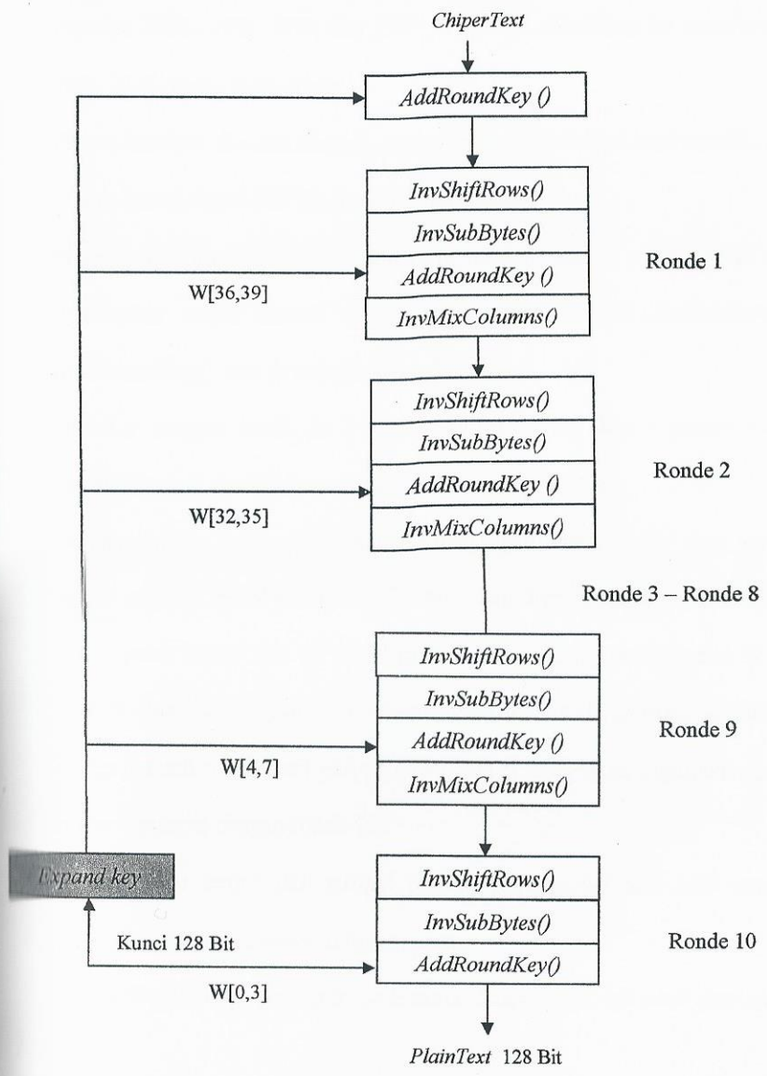
Gambar 3.12 Diagram Proses Enkripsi Rijndael

Dari diagram diatas dapat digambarkan melalui langkah-langkah dibawah

1. Pada awal enkripsi, input *plaintext* 128 bit berupa file citra dengan ekstensi JPEG/ JPG, GIF, dan PNG yang akan disalinkan ke suatu *array* yang diberi nama *state array*.
2. Proses enkripsi dimulai dengan suatu proses yang disebut *AddroundKey*().
3. Untuk kunci *chiper* 128 bit, jumlah ronde adalah 10.
4. Proses selanjutnya adalah 9 ronde yang masing-masing ronde terdiri dari urut-urutan empat macam proses yaitu *SubBytes(), ShiftRows(), MixColumns(),* dan *AddRoundKey().*
5. Diakhiri dengan ronde ke 10 yang hanya terdiri dari 3 proses yaitu *SubBytes(), ShiftRows(),* dan *AddRoundKey().*
6. Dengan proses *Expand Key* atau perpanjangan kunci, kunci yang terdiri dari 4 *word* (1 *word* terdiri dari 32 bit) akan diperpanjang menjadi suatu *array* yang terdiri dari 44 *word* yang masing-masing terisiri dari 32 bit atau 4 *byte*. Kunci yang asli digunakan pada awal enkripsi, kemudian setiap 4 buah *word* hasil perpanjangan kunci yang akan digunakan pada ronde 1 sampai dengan ronde 10.
7. Setiap ronde terdiri dari tempat proses, yang terdiri dari satu operasi permutasi dan tiga operasi subtitusi yaitu:
   a. *SubBytes(),* menggunakan *S-Box* untuk subtitusi *byte* per *byte.*
   b. *ShiftRows(),* permutasi sederhana dengan cara *left shift* atau geser kiri.
   c. *MixColumns(),* suatu operasi aritmatika dalam GF(28).
   d. *AddRoundKey(),* operasi *XOR* antara hasil dari tiga proses sebelumnya dengan 4 *word* kunci Ronde.

3.1.3.3 Proses Dekripsi

Berikut penjelasan proses dekripsi melalui diagram :



Gambar 3.13 Diagram Proses Dekripsi Rijndael

Dari diagram diatas dapat digambarkan melalui langkah-langkah dibawah ini:

1. Pada awal dekripsi, input *Chipertext* 128 bit berupa file citra dengan ekstensi JPEG/ JPG, GIF, dan PNG yang akan disalinkan ke suatu *array* yang diberi nama *state array*.
2. Proses dekripsi dimulai dengan suatu proses yang disebut *AddroundKey()*.
3. Untuk kunci *chiper* 128 bit, jumlah ronde adalah 10.
4. Proses selanjutnya adalah 9 ronde yang masing-masing ronde terdiri dari urut-urutan empat macam proses yaitu *invShiftRows(), InvSubBytes(), AddRoundKey(), dan InvMixColumns()*.
5. Diakhiri dengan ronde ke 10 yang hanya terdiri dari 3 proses yaitu *invShiftRows(), InvSubBytes(),* dan *AddRoundKey()*.
6. Dengan proses *Expand Key* atau perpanjangan kunci, kunci yang terdiri dari 4 *word* (1 *word* terdiri dari 32 bit) akan diperpanjang menjadi suatu *array* yang terdiri dari 44 *word* yang masing-masing terisiri dari 32 bit atau 4 *byte*. Kunci yang asli digunakan pada awal dekripsi, kemudian setiap 4 buah *word* hasil perpanjangan kunci yang akan digunakan pada ronde 1 sampai dengan ronde 10.
7. Setiap ronde terdiri dari tempat proses, yang terdiri dari satu operasi permutasi dan tiga operasi subtitusi yaitu:
   a. *invShiftRows()*, permutasi sederhana dengan cara *left shift* atau geser kiri.
   b. *invSubBytes()*, menggunakan *invS-Box* untuk subtitusi *byte* per *byte*.

c. *AddRoundKey()*, operasi *XOR* antara hasil dari dua proses sebelumnya dengan 4 *word* kunci Ronde.

d. *invMixColumns()*, suatu operasi aritmatika dalam GF(28).

Urutan proses dekripsi Rijndael (AES) tidak merupakan kebalikan dari enkripsi. Proses Dekripsi Rijndael (AES) ini digambarkan pada gambar 28. Ada proses yang dipertukarkan urutannya, walaupun penggunaan kuncinya sama.

Keempat proses dekripsi yang terdapat pada diagram tersebut adalah masing-masing invers dari empat proses pada enkripsi tetapi urutan *InvShiftRows()* dan *InvSubBytes()* dipertukarkan dan *AddRoundKey()* dan *InvMixColumns()* juga dipertukarkan.

3.1.4 Analisis Keamanan *File* Citra

Dalam aplikasi yang dibangun, proses enkripsi dilakukan pada saat membuka *file* sehingga informasi yang ada didalam *file* citra tersebut berupa chipertext atau *file* yang telah tersandikan. Setelah *file* yang disandikan dibuka, *file* tersebut akan tersimpan didalam sebuah folder penyimpanan. Apabila ada orang lain yang bukan pemilik *file* tersebut ingin membukanya, maka akan mendapatkan informasi berupa *chipertext*. Kerahasiaan *file* akan tetap terjaga karena tanpa kunci yang benar tidak akan dapat membuka *file* yang sesungguhnya.

3.2 Analisis Kebutuhan Perangkat Lunak

3.2.1 Dekripsi Umum

Perangkat lunak yang dibangun mampu melakukan enkripsi dan dekripsi terhadap *file* citra. Sebelum *file* tersebut dibuka, terlebih dahulu dilakukan enkripsi untuk menjaga kerahasiaan *file*.

Data-data yang bekerja dalam sistem perangkat lunak yang dibangun yaitu:

1. *Plaintext*

   Plaintext merupakan *file* sesungguhnya.

2. *Chipertext*

   Chipertext merupakan *file* yang telah disandikan menjadi bentuk yang tidak terbaca.

3. Kunci Ekspansi

   Kunci ekspansi merupakan data yang digunakan untuk mengenkripsi dan mendekripsi pesan.

3.2.2 Spesifikasi Kebutuhan Perangkat Lunak

Analisa yang dilakukan sebelumnya akan disesuaikan dengan spesifikasi kebutuhan perangkat lunak yang akan dibahas dalam subbab ini. Fitur-fitur utama yang disediakan perangkat lunak antara lain:

1. Fitur Enkripsi

   Perangkat lunak yang dibangun memiliki fitur enkripsi untuk memulai tahapan enkripsi *file* citra. Didalam fitur *Encrypt* ini terdapat *button*

Browse dan *button* Enkripsi *Button Browse* berfungsi untuk membuka *file* citra didalam *folder* yang akan diamankan. Button Enkripsi berfungsi untuk menjalankan enkripsi setelah memasukkan *key* yang diinginkan dan menyimpan langsung *file* citra ke dalam *folder* penyimpanan.

2. Fitur Dekripsi

   Perangkat lunak yang dibangun memiliki fitur Dekripsi untuk memulai tahapan dekripsi *file* citra. Didalam fitur Dekripsi ini terdapat *button Browse* dan *button* Dekripsi. *Button Browse* berfungsi untuk membuka *file* citra yang telah terenkrip sebelumnya didalam *folder*. Button Dekripsi berfungsi untuk menjalankan dekripsi setelah memasukkan *key* yang asli dan menyimpan langsung *file* citra ke dalam *folder* penyimpanan.

Fitur-fitur diatas akan didapat dengan baik dan benar apabila kebutuhan sistem terpenuhi, baik kebutuhan fungsional maupun kebutuhan non-fungsional. Berikut kebutuhan fungsional perangkat lunak:

1. Sistem dapat membuka *file* apabila *file* yang diambil dari dalam *folder* adalah *file* citra yang berekstensi JPG, JPEG, PNG, dan GIF.
2. Sistem dapat melakukan enkripsi dengan menggunakan algoritma Rijndael (AES) dengan menggunakan kunci.
3. Sistem dapat melakukan dekripsi dengan menggunakan algoritma Rijndael (AES) dengan menggunakan kunci yang asli.
4. Sistem menyediakan kunci untuk enkripsi dan dekripsi dengan kunci yang sama.

5. Sistem dapat menyimpan *file* yang dienkripsi dan didekripsi dalam bentuk *file* citra yang berekstensi JPG/ JPEG, PNG, dan GIF.

Selain kebutuhan fungsional diatas terdapat kebutuhan non-fungsional perangkat lunak yaitu sistem yang dibangun memiliki *interface* yang menarik dan mudah dimengerti oleh *user*.

3.2.3 Model *Use Case*

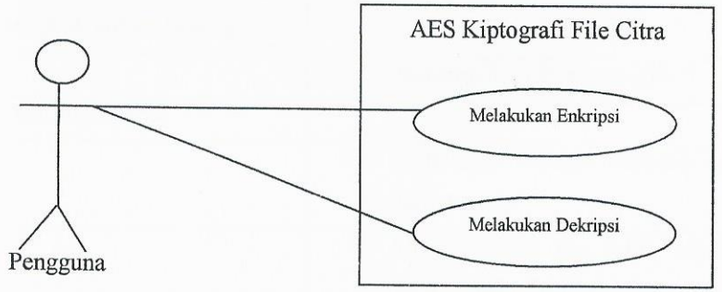
Pada subbab ini akan dijelaskan gambaran fungsionalitas perangkat lunak yang dibangun dengan menggunakan pemodelan Use Case.

3.2.3.1 Aktor dan Tujuan

Tabel 3.1 Aktor dan Tujuan

Aktor	Nama *Use Case*	Tujuan *Use Case*
Pengguna	*Encrypt*	Mengenkripsi *file*
	Decrypt	Mendekripsi *file*

3.2.3.2 Diagram *Use Case*



Gambar 3.14 Diagram *Use Case*

3.2.3.3 Skenario *Use Case*

1. No : 001

   Nama *Use Case* : Enkripsi

   Aktor : Pengguna

   Tujuan : mengenkripsi *file*

   Dekripsi : *Use Case* ini digunakan oleh Pengguna untuk melakukan enkripsi *file* citra

   Kondisi Awal : Pengguna belum mengenkripsi *file*

   Kondisi Akhir : Pengguna telah mengenkripsi *file* dan menyimpan *file* tersebut ke dalam *folder*

Tabel 3.2 Skenario *Use Case* Enkripsi

Aktor	Sistem
1. Memilih menu Enkripsi	
	2. Menampilkan *form* Citra *File Encryption*
3. Membuka *file* citra dengan mengklik *button Browse*	
	4. Menampilkan *form open file*
5. Memilih *file* citra	
	6. Melakukan validasi terhadap masukan dari aktor
	7. Memunculkan *file* pada *label form* ketika validasinya benar.
8. Mengentri Kunci (16 byte)	

9. Menekan *button* Enkripsi	
	10. Melakukan validasi terhadap masukan dari aktor. Jika kunci yang dimasukkan lebih dari 16 *byte*, maka akan muncul pesan kesalahan.
11. Mengentri nama untuk *file* yang akan disimpan lalu klik *OK*	
	12. Melakukan penyimpanan *file* yang telah dienkripsi dan memunculkan pesan sukses ketika validasinya benar.

2. No : 002

   Nama *Use Case* : Dekripsi

   Aktor : Pengguna

   Tujuan : mendekripsi *file*

   Dekripsi : *Use Case* ini digunakan oleh Pengguna untuk melakukan dekripsi *file* citra

   Kondisi Awal : Pengguna belum mendekripsi *file*

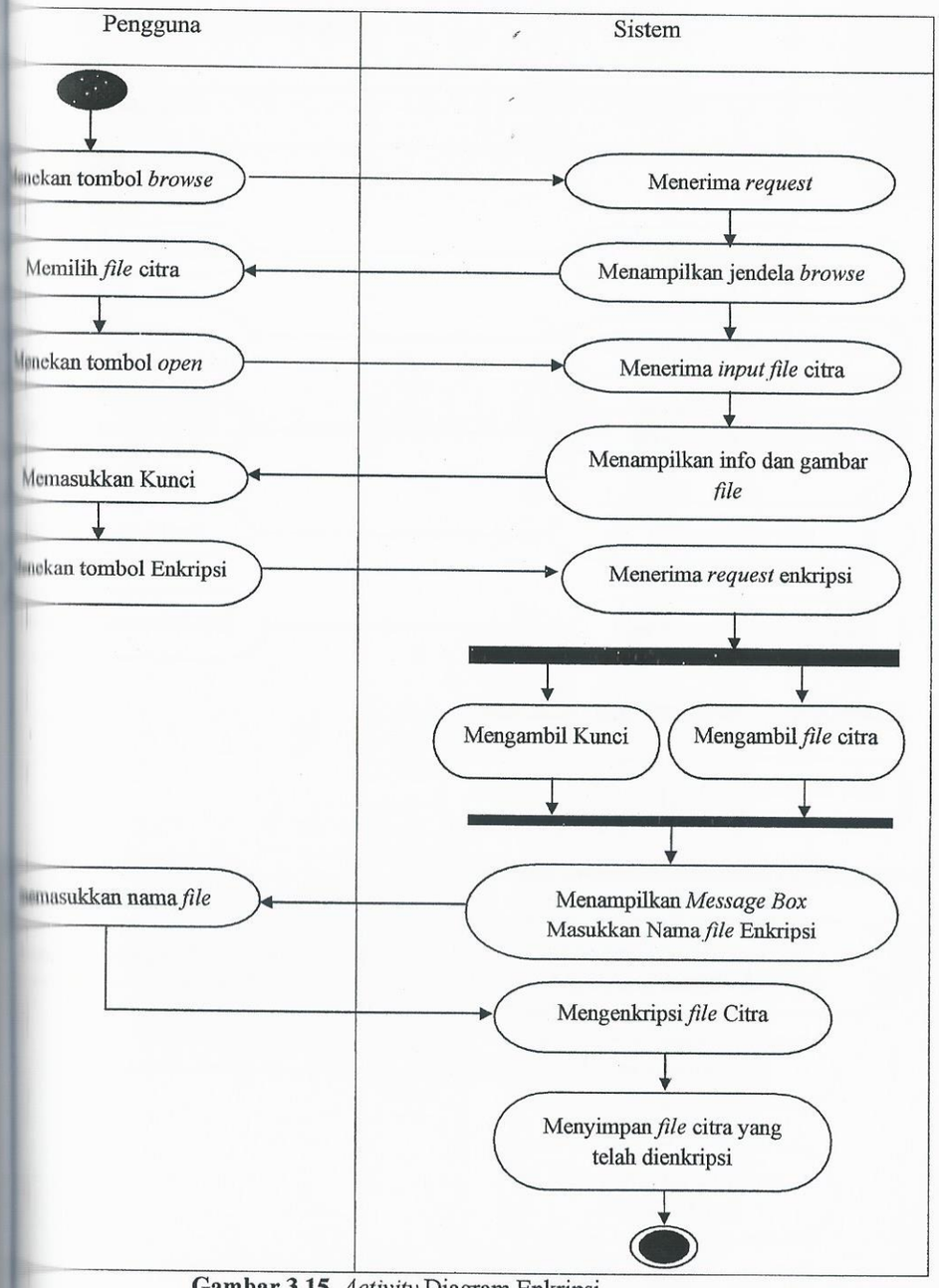
   Kondisi Akhir : Pengguna telah mendekripsi *file* dan menyimpan *file* tersebut ke dalam *folder*

Tabel 3.3 Skenario *Use Case* Dekripsi

Aktor	Sistem
1. Memilih menu Dekripsi	
	2. Menampilkan *form* Citra *File Decryption*
3. Membuka *file* citra dengan mengklik *button Browse*	
	4. Menampilkan *form open file*
5. Memilih *file* citra yang telah dienkripsi	
	6. Melakukan validasi terhadap masukan dari aktor
7. Mengentri Kunci (16 byte) yang asli	
8. Menekan *button* Dekripsi	
	9. Melakukan validasi terhadap masukan dari aktor. Jika kunci yang dimasukkan lebih dari 16 *byte*, maka akan muncul pesan kesalahan.
10. Mengentri nama untuk menyimpan *file* yang didekripsi lalu klik *OK*	
	11. Melakukan penyimpanan *file* yang telah didekripsi dan memunculkan pesan sukses ketika validasinya benar.

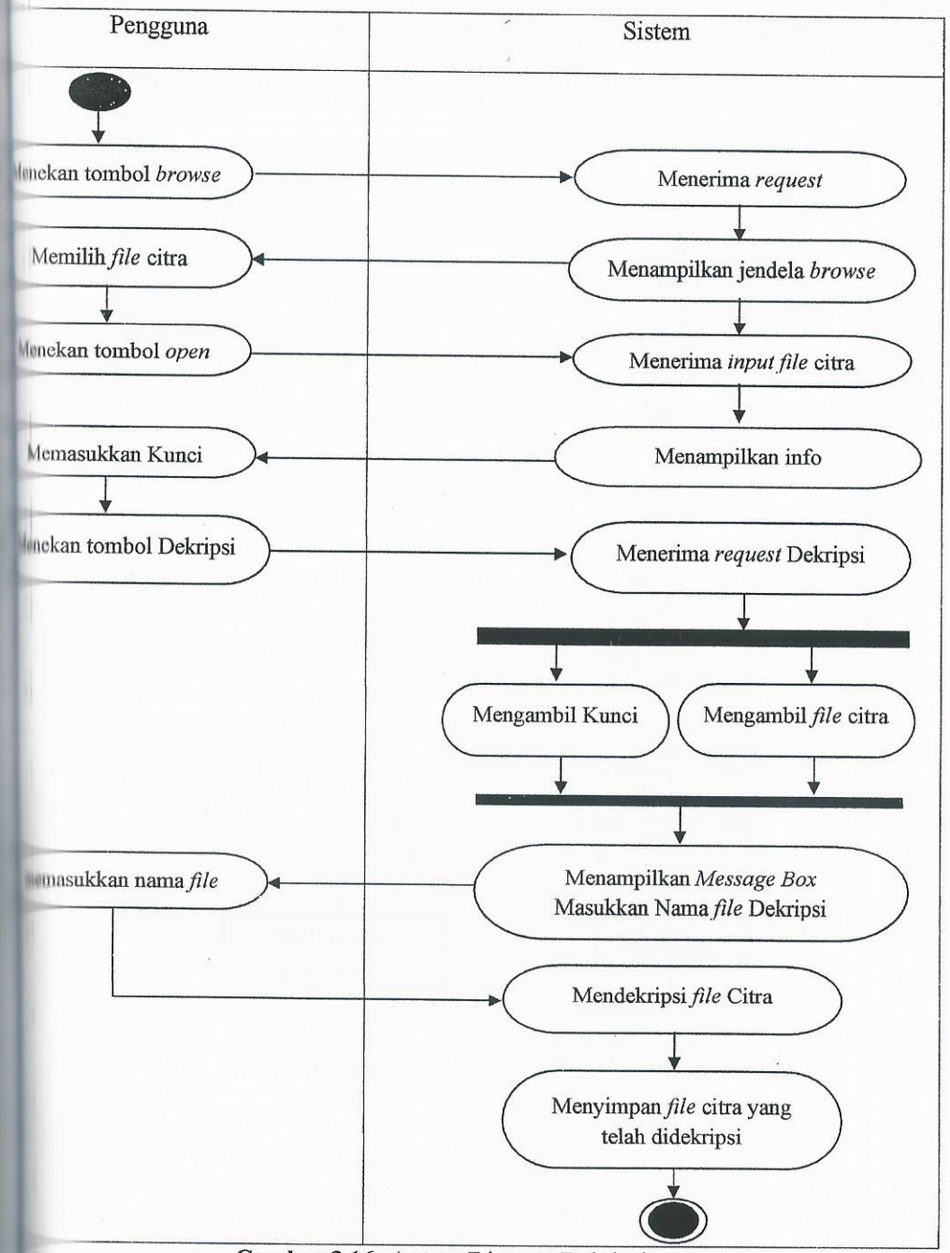
3.2.3.4 *Activity* Diagram

1. *Activity* Diagram Enkripsi



Gambar 3.15 *Activity* Diagram Enkripsi

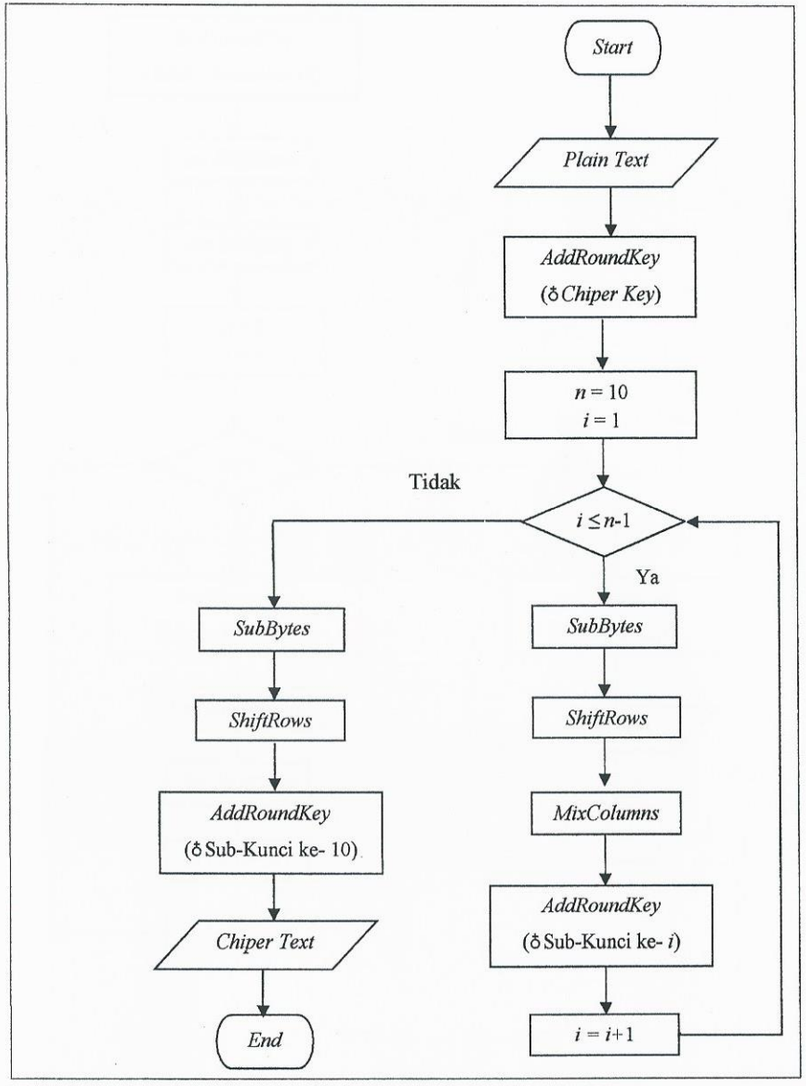
2. *Activity* Diagram Dekripsi



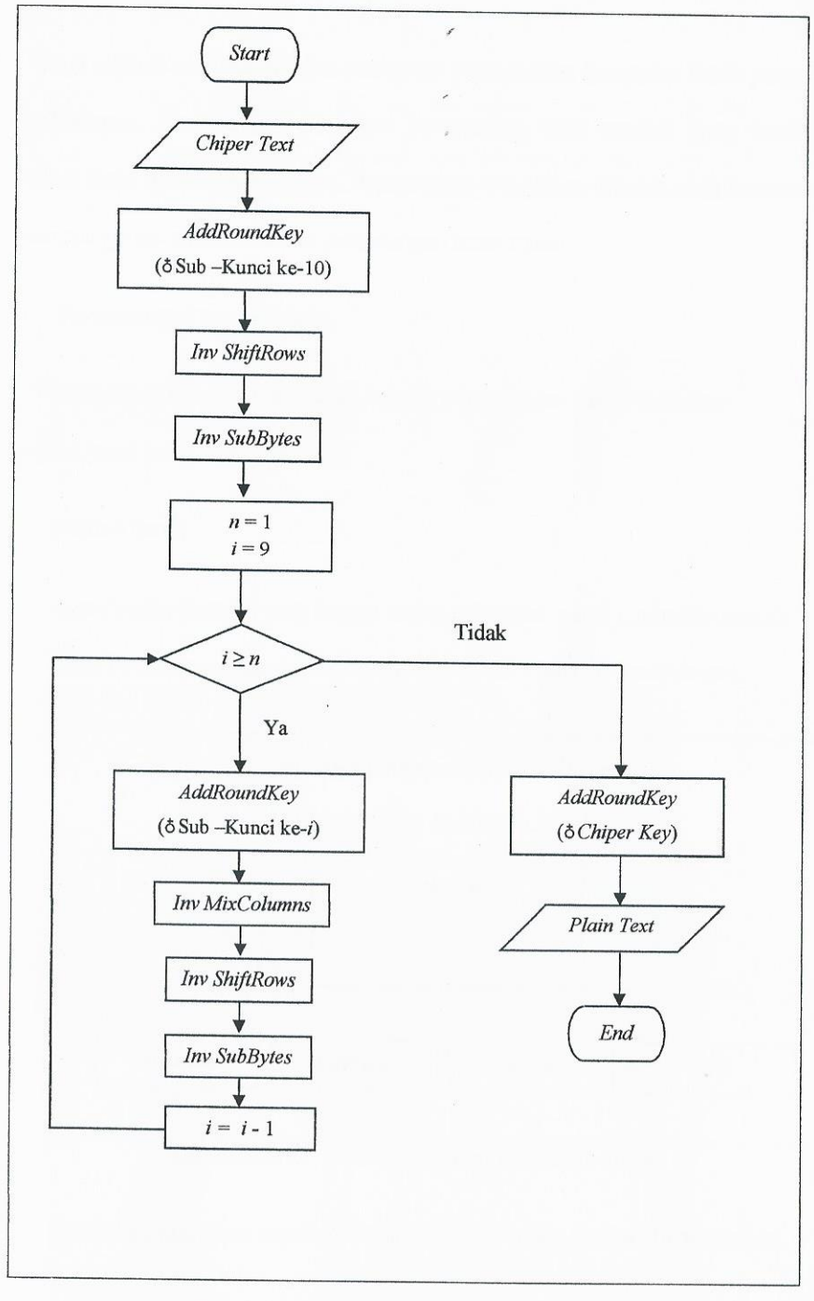
Gambar 3.16 *Activity* Diagram Dekripsi

3.2.4 *Flowchart* Algoritma Rijndael

Adapun *flowchart* algoritma Rijndael pada proses enkripsi dan dekripsi adalah sebagai berikut :



Gambar 3.17 *Flowchart* Proses Enkripsi Algoritma Rijndael



Gambar 3.18 *Flowchart* Proses Dekripsi Algoritma Rijndael

3.3 Perancangan Perangkat Lunak

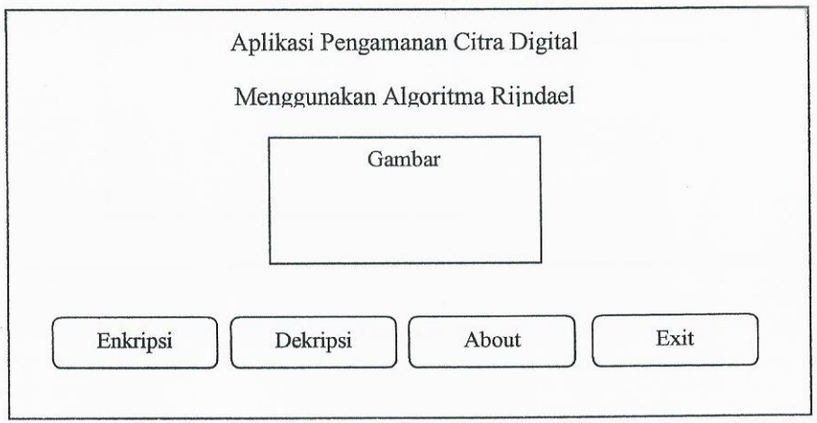
Pada subbab ini akan dibahas mengenai perancangan perangkat lunak yang akan dibangun. Perancangan dilakukan berdasarkan hasil analisis yang telah dilakukan pada subbab sebelumnya. Perancangan yang akan dibahas pada subbab ini meliputi perancangan data dan perancangan antar muka.

3.3.1 Perancangan Antar Muka

Pada subbab ini akan membahas tentang perancangan antar muka dari perangkat lunak yang ingin dibangun.

1. Menu Utama

   Antar muka pertama yang tampil ketika perangkat lunak dijalankan adalah antar muka menu utama yang terdiri dari menu Enkripsi dan Dekripsi.

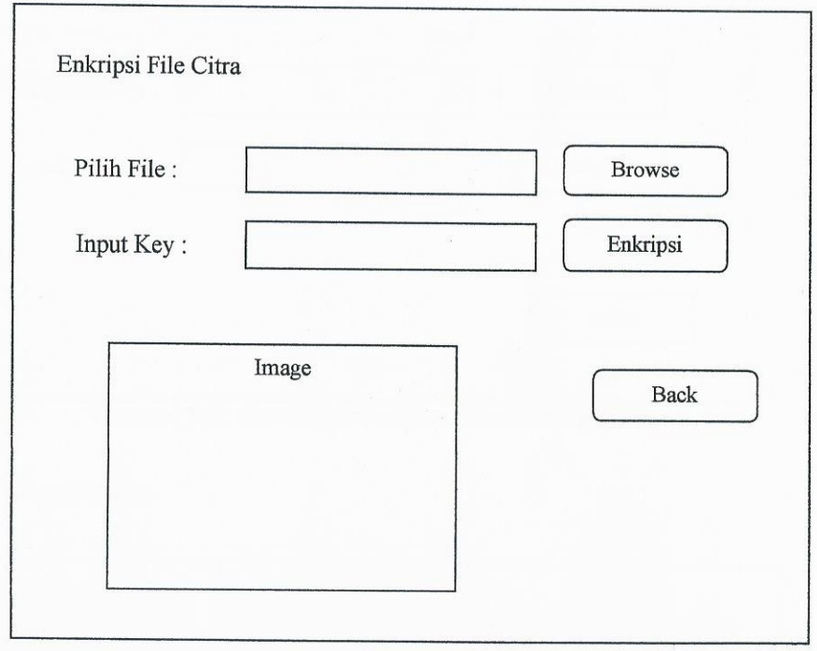


Gambar 3.19 Rancangan Tampilan Menu Utama

Kemudian pengguna dapat memilih salah satu menu yang akan dijalankan.

2. Menu Enkripsi

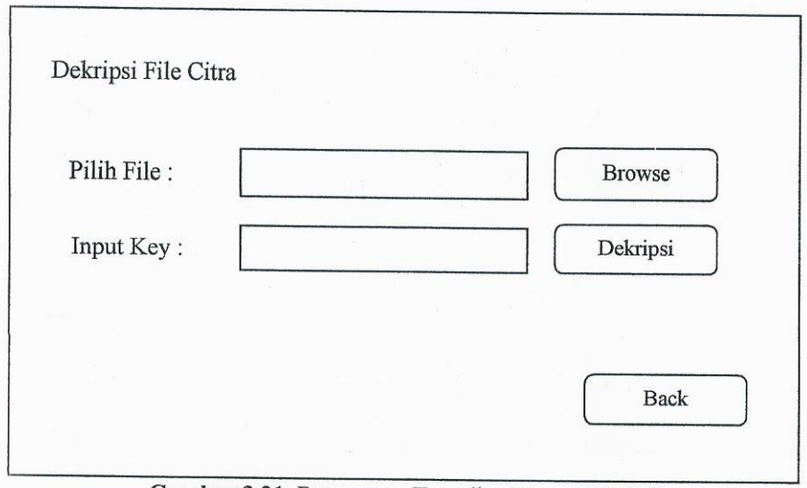
Menu enkripsi digunakan oleh pengguna, ketika menu ini dipilih oleh pengguna, maka akan muncul antar muka *file* citra enkripsi yang digunakan untuk melakukan enkripsi *file*.



Gambar 3.20 Rancangan Tampilan Menu Enkripsi

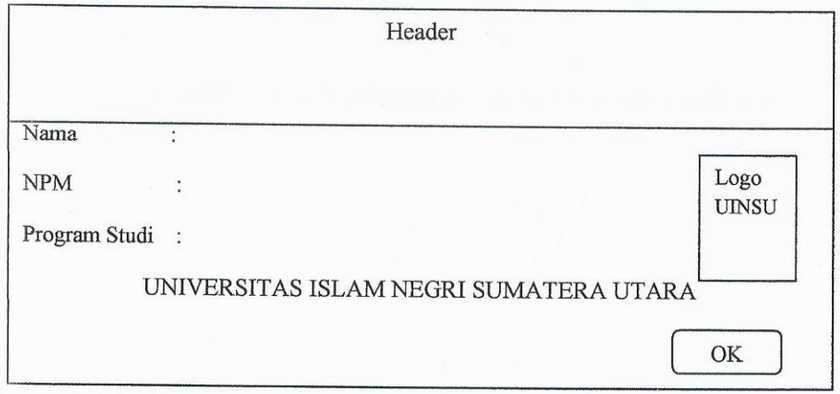
3. Menu Dekripsi

Menu dekripsi digunakan oleh pengguna, ketika menu ini dipilih oleh pengguna, maka akan muncul antar muka *file* citra dekripsi yang digunakan untuk melakukan dekripsi *file*.



Gambar 3.21 Rancangan Tampilan Menu Dekripsi

4. Menu Biodata



Gambar 3.22 Rancangan Tampilan Menu Biodata

BAB IV

IMPLEMENTASI DAN HASIL

4.1 Implementasi

Implementasi merupakan proses yang dilakukan setelah tahapan analisis dan perancangan sistem selesai dilakukan. Tujuan yang dicapai pada tahap ini adalah dapat di operasikannya hasil perancangan sistem yang telah dibuat.

4.1.1 Perangkat Lunak

Perangkat lunak yang harus disiapkan dalam Implementasi Algoritma Rijndael pada Perancangan Aplikasi Pengamanan Citra Digital menggunakan Java dengan menggunakan beberapa *development tools*, antara lain:

1. Netbeans IDE 7.4

   Sebagai media untuk penulisan script Java.

2. Java

   Untuk membuat aplikasi pengamanan citra digital ini menggunakan Java sebagai script.

4.1.2 Perangkat Keras

Spesifikasi perangkat keras yang digunakan penulis dalam pembuatan aplikasi pengamanan citra digital ini adalah sebagai berikut:

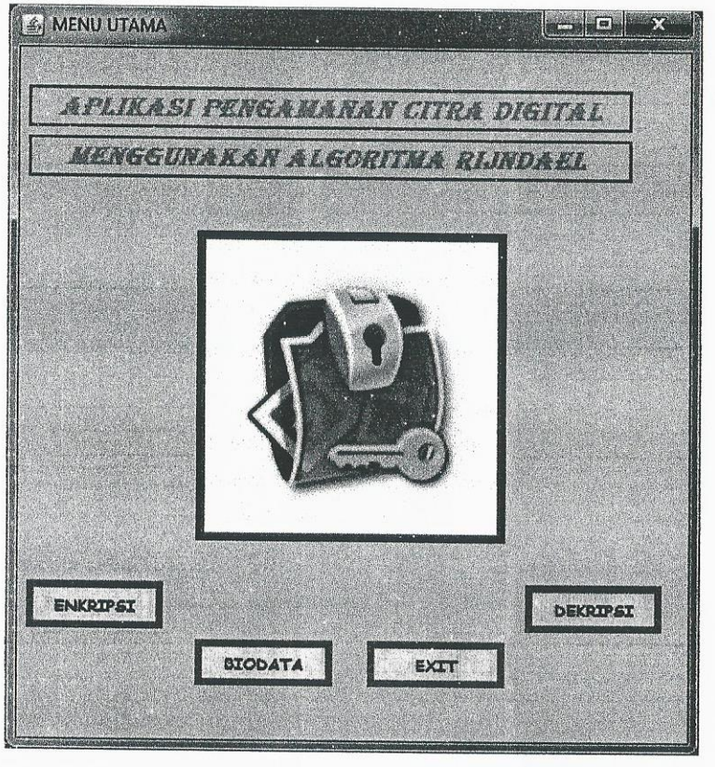
1. *Processor* Intel ® Core™ i3
2. RAM 2 GB
3. VGA dengan kapasitas 762 MB
4. *Harddisk* 500 GB
5. Alat-alat pendukung lain seperti monitor, *mouse* dan *keyboard.*

4.2 Pengujian Aplikasi

Pengujian ini dilakukan untuk memeriksa hasil dari aplikasi yang diimplementasikan. Tujuan utama dari pengetesan aplikasi ini adalah untuk memastikan bahwa elemen-elemen atau komponen-komponen dari aplikasi telah berfungsi sesuai dengan yang diharapkan. Pengujian perlu dilakukan untuk mencari kesalahan yang mungkin masih terjadi. Berikut ini adalah *printscreen* dari program yang telah dibuat:

4.2.1 Tampilan Menu Utama

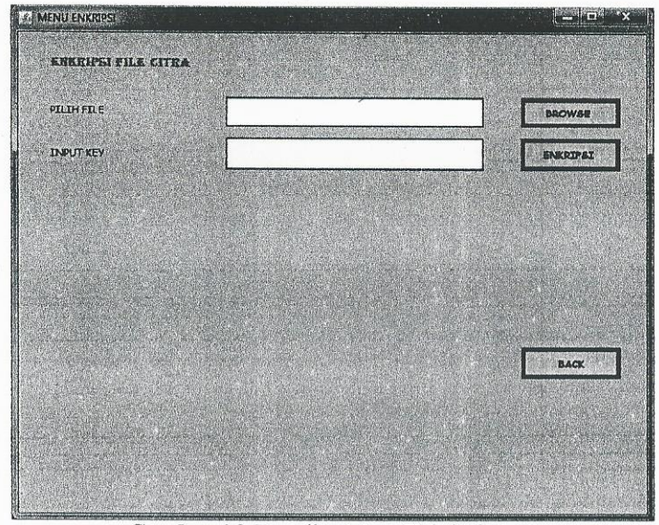
Tampilam menu utama pada aplikasi merupakan tampilan dimana untuk mulai menggunakan aplikasi. Tampilan menu utama ini terdiri dari beberapa submenu, yaitu enkripsi, dekripsi, *about* dan *exit.* Adapun tampilan menu utama yang telah ditesting sebagai berikut:



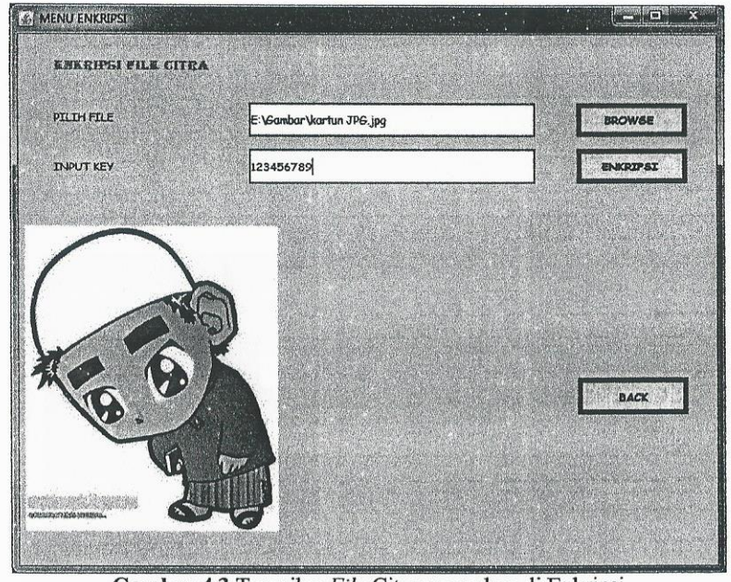
Gambar 4.1 Tampilan Menu Utama

4.2.2 Tampilan Menu Enkripsi

Tampilan menu enkripsi adalah tampilan yang didalamnya terdapat tombol *button browse* untuk memunculkan *open dialog file*, tombol *button* enkripsi untuk mengenkripsi *file* citra yang dipilih, dan tombol *button back* untuk mengembalikan ke menu utama.

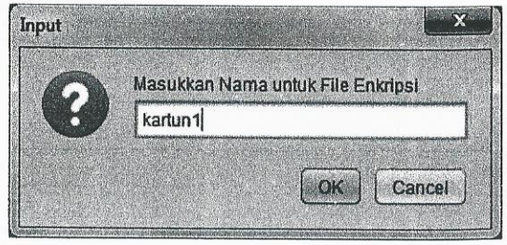


Gambar 4.2 Tampilan Menu Enkripsi



Gambar 4.3 Tampilan *File* Citra yang akan di Enkripsi

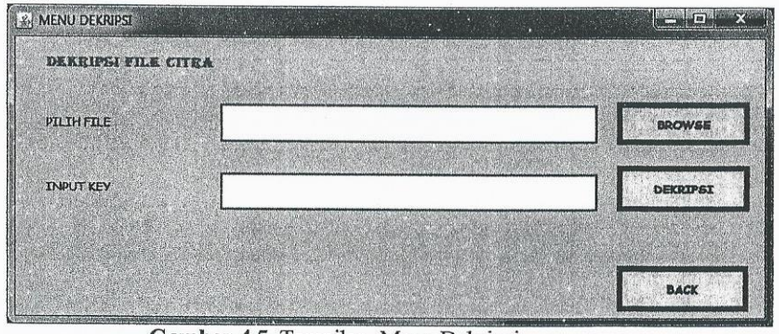
Pada saat tombol enkripsi diklik maka akan muncul *message box*, untuk memasukkan nama file citra yang dienkripsi.



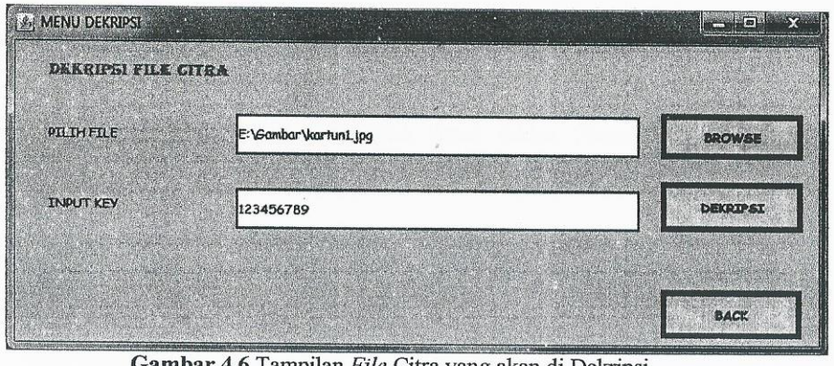
Gambar 4.4 *Message Box* Nama *File* Enkripsi

4.2.3 Tampilan Menu Dekripsi

Tampilan menu dekripsi adalah tampilan yang didalamnya terdapat tombol *button browse* untuk memunculkan *open dialog file*, tombol *button* dekripsi untuk mendekripsi *file* citra yang dipilih, dan tombol *button back* untuk mengembalikan ke menu utama.

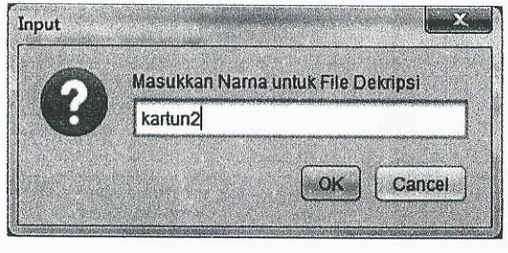


Gambar 4.5 Tampilan Menu Dekripsi



Gambar 4.6 Tampilan *File* Citra yang akan di Dekripsi

Pada saat tombol dekripsi diklik maka akan muncul *message box*, untuk memasukkan nama *file* citra yang didekripsi.



Gambar 4.7 *Message Box* Nama *File* Dekripsi

4.3 Hasil

Hasil dari merancang sebuah aplikasi pengamanan citra digital adalah sebuah program yang berfungsi mengamankan *file* citra digital yang ingin dirahasiakan.

Aplikasi pengamanan *file* citra digital ini terdapat 2 proses yaitu proses enkripsi dan dekripsi. Dimana proses enkripsi pada *file* citra yaitu mengubah citra digital menjadi citra digital yang tidak dapat dilihat (diamankan) dengan memberikan kunci pada proses enkripsi. Proses dekripsi *file* citra mengubah *file* citra yang tidak dapat dilihat(diamankan) kembali ke bentuk semula dengan memasukkan kunci yang sesuai dengan proses enkripsi. Apabila user memasukkan kunci yang tidak sesuai dengan proses enkripsi maka citra tidak akan dapat dilihat.

Aplikasi pengamanan *file* citra digital ini dapat digunakan pada komputer umum seperti komputer umum yang terdapat di Laboratorium komputer Sekolah dan Universitas, agar gambar yang bersifat rahasia / pribadi tidak dapat dilihat oleh pihak yang tidak berkepentingan.

4.3.1 Kelebihan Dan Kekurangan Aplikasi

1. Kelebihan
   - Dapat menjaga keamanan citra digital
   - Dapat menggunakan jenis kunci *input* bervariasi
   - Dapat mengamanankan *file* citra dengan format *file* JPEG/JPG, PNG dan GIF
   - *File* citra yang telah dienkripsi dapat dienkripsi kembali dan didekripsi sesuai dengan enkripsi yang dilakukan.

2. Kekurangan

   - Tidak dapat menyimpan *file* citra hasil dekripsi dengan nama dan tempat yang sama dengan *file* citra hasil enkripsi.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil pembuatan dan implementasi yang telah dilakukan guna penyusunan laporan penelitian berjudul " Perancangan Aplikasi Pengamanan Citra Digital Menggunakan Java Dengan Algoritma Rijndael " maka dapat diambil kesimpulan sebagai berikut:

1. Aplikasi yang dibuat adalah aplikasi pengamanan citra digital dengan menggunakan Algoritma Rijndael yang digunakan untuk membantu menjaga keamanan *file* citra digital dari pihak yang tidak berkepentingan.
2. Aplikasi pengamanan citra digital ini dibangun dengan menggunakan bahasa pemrograman Java.
3. Aplikasi ini dapat melakukan proses enkripsi dan dekripsi *file* citra dengan format *file* citra JPEG/JPG, PNG dan GIF.
4. Panjang kunci dan ukuran blok pada algoritma Rijndael dapat dipilih secara independen karena Algoritma Rijndael mendukung panjang kunci 128 bit sampai 256 bit dengan step 32 bit. Pada penelitian ini panjang kunci dan ukuran blok yang digunakan adalah 128 bit.

5.2 Saran

Adapun saran dari hasil perancangan program pengamanan citra digital menggunakan Java ini adalah :

1. Ukuran kunci yang digunakan dalam proses enkripsi dan dekripsi untuk selanjutnya dapat dikembangkan dengan menggunakan ukuran 192 bit atau 256 bit.
2. Bahasa pemrograman yang digunakan untuk perancangan program pengamanan citra digital dengan algoritma Rijndael ini masih menggunakan bahasa pemrograman Java. Oleh karena itu untuk pengembangan lebih lanjut disarankan dapat menjadikannya program yang berbasis Android.

DAFTAR PUSTAKA

Ariyus, Dony. 2008. *Pengantar Ilmu Kriptografi Teori Analisis dan Implementasi.* Yogyakarta: Andi Offset.

A.S, Rosa, dan M. Shalahuddin. 2014. *Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek.* Bandung: Informatika.

Putra, Darma. 2010. *Pengolahan Citra Digital.* Yogyakarta: Andi

Al Fatta, Hanif. 2007. *Analisis dan Perancangan dan Sistem Informasi untuk Keunggulan Bersaing Perusahaan dan Organisasi Modern.* Yogyakarta: Andi.

Dwi Prasetyo, Didik. 2007. *150 Rahasia Pemrograman Java.* Jakarta: Elex Media Komputindo.

Farid Fachrurozi, Muhamad. 2006. *Enkripsi pesan Rahasia Menggunakan Algoritma (Advanced Encryption Standard) AES: Rijndael.* Jakarta: Universitas Islam Negeri Syarif Hidayatullah.

Yolanda S, Elfira. 2008. *Implementasi disk Encryption Menggunakan Algoritma Rijndael.* Bandung: Institut Teknologi Bandung.

Satria, Eko. 2009. *Studi Algoritma Rijndael Dalam Sistem Keamanan Data.* Medan: Universitas Sumatera Utara.

Sujatmiko, Edy. 2007. *Pemilihan Algoritma Optimal Untuk Kompresi Data Citra Iris Mata Manusia.* Semarang: Universitas Diponegoro.

Seftiani, AR. 2012. *Analisis Kualitas Visual pada Hasil Citra Kompresi Dengan Menggunakan Metode Run Length Encoding (RLE).* Medan: Universitas Sumatera Utara.